COMMODORE USERS

48355
K

Ahoy!

$2.50/CAN. $2.75 OCT. 1984



Ahoy!

CONTENTS

DEPARTMENTS

FEATURES

PROGRAMS

Cover illustration by James Regan
(Illustrations inside screens by Art Huff, Kat Reviaska, JoAnn Case, and James Regan)

Publisher
Michael Schneider

Editor
David Allikas

Managing Editor
Robert J. Sodaro

Senior Editor
Tim Moriarty

Technical Editor
David Barron

Consulting Editors
Ben Bova
Morton Kevelson
Dale Rupert

Art Director
Raoul Tenazas

Associate Art Director
JoAnn Case

Art Production
Christopher Carter
Pauline Giordani
Eve Griffin

Technical Advisors
Edward D. Laufer
Bob Lloret

Circulation Director
W. Charles Squires

Advertising Director
Lynne Dominick

Director of Promotion
Joyce K. Fuchs

Controller
Dan Tunick

Managing Director
Richard Stevens

Advertising Representative
JE Publishers Representative
6855 Santa Monica Blvd., Suite 200
Los Angeles, CA 90038
(213)467-2266
Boston (617)437-7628
Dallas (214)660-2253
New York (212)724-7767
Chicago (312)445-2489
Denver (303)595-4331
San Francisco (415)864-3252

AHOY! No. 10, October 1984. Published monthly by Ion International Inc., 45 W. 34th St., Suite 407, New York, NY, 10001. Subscription rate: 12 issues for $19.95, 24 issues for $37.95 (Canada $26.95 and $49.95 respectively). Application to mail at 2nd Class postage rates is pending at New York, NY and additional mailing offices.

Postmaster, send address changes to *Ahoy!*, 45 W. 34th Street, Suite 407, New York, NY 10001. All editorial and subscription inquiries and software and hardware to be reviewed should be sent to 45 W. 34th St., Suite 407, New York, NY 10001.

VIEW FROM THE BRIDGE

It's a natural: *Ahoy!*, the most graphically appealing Commodore magazine on the market, devoting an issue to Commodore graphics! This month features only half the exhaustive treatment afforded the subject by Morton Kevelson, including a look at bit mapped graphics and reviews of some of the top packages available. His two-part feature concludes next month with...but we'll let you wait and be surprised.

In the meantime, Morton's guest editorial—though deserving of inclusion in our editor's page—forces us to talk more sketchily than usual about the contents of the October *Ahoy!*:

Pete Lobl is back! The author of *Multi Draw 64* and the *Interrupt Music Maker/Editor* has provided the *VIC 40-Column Operating System*, enabling VIC users to generate 40 columns and imbue their computers with many of the characteristics of a PET. (Turn to page 45.)

Part II of Orson Scott Card's feature on programming text adventures appears in *Creating Your Own Games on the VIC and 64*, including the complete, ready to play *Emerald Elephant of Cipangu*. (Turn to page 54.)

Not content with covering Commodore graphics like a ballpark tarpaulin, Morton the K has laid bare the 1541's Block Availability Map in *BAM Read and Print*. (Turn to page 53.)

There's lots more inside, like Michael Buhidar's entertaining *Lawn Job* for the C-64 (turn to page 11); the windup of Richard Herring's acclaimed *Educational Software* series, delayed from a previous issue (turn to page 62); Dale Rupert's report on using your *Joystick!* (yes, joystick!) in your programming (turn to page 19); and many other surprises.

One last note: remember that our programs are now available on disk. See the ad on page 66.

Thanks for shipping with us once again!

THE GRAPHICS CHALLENGE

By Morton Kevelson

The other day an acquaintance presented me with a terrific new program he had developed. It would allow the owner of a personal computer to manipulate text to his heart's content. It included many powerful features such as the ability to move, insert, replace, add, delete, and so on. He called his new idea a "word processor." Well, I started to work with it and sure enough it did everything he said it would. Before long, I was composing error-free letters at a phenomenal rate. Unfortunately, the program had one minor restriction. It would only accept a single screen of text at a time....

Sounds a little farfetched, does it not? None of us would seriously accept my fictional friend's proposal for a single screen word processor. We all know that such a program would be severely limited in its scope and application. The analogy may be somewhat extreme, but at some point during the past month it dawned on me that I had been asked to accept just such a proposal. This was in regard to the "graphics processors" reviewed in this issue.

Now don't get me wrong, these packages were very impressive. They certainly turned the Commodore 64 into a very effective sketchpad. However, it seems that a significant opportunity has been missed. None of these programs allowed more than a single screen image to be created at a single time. It was as if the world were limited to the 160 by 200 (320 by 200 in hi-res mode) pixels available to a single screen display. Not one program allowed horizontal or vertical scrolling of the screen, or even linking of image files from disk.

In this regard, I would like to present a challenge to all the software developers out there. Give us a true "graphics processor." Show us what the Commodore 64 can really do. Let us do horizontal and vertical scrolling for some real panoramic displays. Sure, graphics eats up a lot of memory at 10,001 bytes for a multicolor screen. Even with 64K to work with, things will get a little tight. How about linked files, where graphics data is stored on the disk and loaded in for a dynamic display? That would be one way to get around the memory crunch.

While we are at it, let us throw in a few more features. A package which would give the user the option of high resolution or multicolor graphics would be a welcome addition. Not one of the reviewed programs had this obvious capability. Let us take this concept one step further. The VIC II chip's line scan interrupts should allow the mixing of hi-res and multicolor graphics on the same screen. Now that would be an impressive display!

So let's get to work and make a real graphics processor for the Commodore 64.

SCUTTLEBUTT

TELECOMMUNICATIONS UPDATE • EXPANSION INTERFACE • DVORAK KEYBOARD • BUSINESS GRAPHICS • COMPUTER LOCK • HANDICAPPING PROGRAM • BASIC EXPANDER • TINK!TONK! AND MR. PIXEL • FINANCIAL PLANNING • LATEST GAME RELEASES FOR THE VIC AND 64

TELEPHONE TIME

You may have noticed that *Scuttlebutt* has been skimpy on telecommunications in the past. But no more! Below is the latest telecom news for you to download, to be followed by a regular telecommunications column beginning in an upcoming issue.

Prentice-Hall's *SkiWriter II* is a word processor with one handy addition: a built-in terminal package. This allows a user to call up a database in the middle of editing a document, download needed information, and incorporate this data into his document. Available in September for the C-64; price $69.95.

Prentice-Hall, P.O. Box 819, Englewood Cliffs, NJ 07632 (phone: 201-592-2611).

Shrinks are the latest casualty of the computer age. *Telepsych,* run by Dr. Timothy Miller, offers psychological counseling to those who have a computer and modem. The client prepares a document describing his problem and uploads it to the *Telepsych* system. Within seven days Dr. Miller replies by posting a letter for the client to receive via the computer system. Talk is neither cheap in person nor by computer, though: the rate is $\frac{7}{10}$ of a cent for each word you type and 2 cents for each word of Dr. Miller's reply.

The lack of face-to-face interaction is a disadvantage of this type of counseling. But on the positive side, composing a letter describing your problems helps you in thinking them over. The modem number for *Telepsych* is 209-473-8296.



Banking via modem (with Chemical Bank's Pronto*) beats standing in line.*
READER SERVICE NO. 236

Chemical Bank has made its *Pronto* home banking system available to Commodore users. *Pronto* allows a user to pay bills, transfer funds, determine his balance, see an electronic statement, track a budget, balance his checkbook, and find out if a check has cleared. Additionally, users can send electronic mail to each other via the system.

Pronto also offers information services, including economics, business, taxes, *Consumer Reports* articles, and a guide to all Chemical Bank services. To be introduced this fall are investment, stock and option training, and a special *Pronto* for small businesses.

Pronto is designed to operate on a Commodore 64 with a 1541 drive and a modem. The Vicmodem, Automodem, and Hesmodem are currently supported. Supplied is a diskette with the *Pronto* terminal software and a speed copy utility for the 1541.

For information call toll-free: 1-800-782-1100.

It seems that everyone is getting into the computer-aided investment act, including *The Source.* In conjunction with Spear Securities Inc., *The Source* will offer online securities trading and confirmation, real-time and delayed stock quotations, automatic portfolio updating and record keeping, and a wide range of investment databases. Those who wish to actively trade online must have an account with Spear Securities.

Source Telecomputing Corporation, 1616 Anderson Road, McLean, VA 22102 (phone: 703-734-7500).

Time to lose carrier for another month. Stay online for our upcoming column.

PLACE YOUR BITS

Having announced *Speed Handicapper* in the June *Scuttlebutt,* it was a probable 12 to 7 that we'd soon hear from manufacturers of other 64-compatible horse race programs. 3G Company has come in at the head of the pack with *Horses OTB.*

When fed statistics from the daily racing form, *Horses OTB* will derive odds for each horse. No judgment is necessary. One advantage over track odds is that the programs will not reflect the amount of money bet on each horse.

3G Company, Inc., Rte. 3, Box 28A, Gaston, OR 97119 (phone: 503-662-4492).

POWER/PAC

Two new 64 utilities from Educomp:

Power Plus ($19.95) adds 40 commands to the 64, including ones for automatic line numbers, screen dump, renumber, find/replace, color change, machine language, and more. (One interesting feature is Un-New, which lets users who have a reset switch reset their computer without losing a program in memory.)

Disk Pac ($14.95) can perform such services as undeleting files, changing a disk ID and name, inspecting all sectors on a disk, and reading the starting and ending addresses of a file.

Educomp, 2139 Newcastle Avenue, Cardiff, CA 92007 (phone: 619-942-3838).

PRACTINEWS

PractiFile ($55) is a database program designed by Practicorp to complement their *PS* and *PractiCalc 64* spreadsheets. It allows files to be merged from several word processing programs and can be used to keep track of mailing lists, accounts receivable, grade book, or other database applications. Over 1000 mailing list records can be stored.

Practicorp, The Silk Mill, Newton Upper Falls, MA 02164 (phone: 617-965-9870).

COMMODORE ANCHOR

If you're afraid your Commodore might sail away one day, try Technalock by Business Security Systems. The $24.95 package consists of two plates which you bond to your computer and the surface you wish to secure it to, a cable that's fed through loops in the plates, lock, and surface cleaner. A package of two additional plates to accommodate peripherals may be purchased for $14.95.

Business Security Systems, 512 South Hanley, Suite 141, St. Louis, MO 63105 (phone: 314-962-4446).

BUSINESS GRAPHICS

B/Graph by Commodore will take any raw data and convert it into full color, 3-dimensional charts, graphs, pie charts, histograms, and other business graphics. Graphs created may be printed on a Commodore MPS 801 printer. Different colors, multiple graphs, and grid overlays are all provided for. For the C-64, the price will be in the $59-$79 range.

Commodore Business Machines, 1200 Wilson Drive, West Chester, PA 19380 (phone: 215-431-9100).

EDUCATIONAL SOFTWARE

No starting on a shoestring for Mindscape, Inc., which has entered the educational market with a full-blown line of software. (It's easy to be bold when you're a subsidiary of SFN, the nation's largest children's textbook publisher.) Mindscape's programs will come from independent developers and be marketed under several categories.

The Pixelwerks series for children 8-12 develops a variety of skills. *Keyboard Cadet* teaches kids both to type fast and to employ good typing technique. *Show Director* lets children write a script and use graphics, animation, and music to stage it. A division of the division is the *Mr. Pixel* series, which allows kids to draw, design, and animate cartoons within a programming framework. First releases are *Mr. Pixel's Electric Paint Set* and *Mr. Pixel's Cartoon Kit.*

Highlighting the Sprout series for kids 4-8 are eight *Tink!Tonk!* programs by noted children's author Mercer Mayer. The first ones available are *Tink's Adventure* (ABC's), *Tinka's Mazes* (addition), *Tuk Goes to Town* (spelling and vocabulary), and *Tonk in the Land of Buddy-Bots* (shape and pattern recognition).

The first release in the Teen/Adult line is *Crossword Magic,* which allows you to generate your own; and in the Productivity/Utility line, *The Perfect Score,* an SAT preparation program.

And finally (for now), Mindscape and the Bank Street College will produce the Bank Street series of educational programs. The *Bank Street Musicwriter* ($49.95) lets you arrange notes on two on-screen staffs, program and play four voices at once, store up to 8000 notes, and compose with notes from whole to 32nd's. The *Bank Street Storybook* ($39.95), available sometime after October, lets the user draw pictures on the screen, edit and color them, and add story text.

Mindscape, Inc., 3444 Dundee Road, Northbrook, IL 60062 (phone: 312-480-7667).



Mindscape's **Show Director** *lets children put together a theater production.*
READER SERVICE NO. 233

Reader Service No. 234

VIC AND 64 GAMES

Atarisoft has begun releasing all-new games and educational software for the C-64 and VIC 20.

Gremlins requires you, like the hero of the movie, to capture all the darling furballs before they touch water and multiply, or eat and transform into deadly monsters. For the C-64.

Typo Attack pits the hunt-and-pecker against waves of descending letters. For the 64 and VIC.

Atarisoft continues to adapt arcade successes to Commodore format, the latest being *Track and Field* (six Olympic events, with a free three-button arcadelike controller in each package), *Donkey Kong, Jr.* (the arcade ape's son tries to rescue his trapped papa), *Mario Brothers* (the *Donkey Kong* hero and his brother versus various forms of pseudolife in dark underground passageways), and *Crystal Castles* (Bentley Bear cases a 3D castle in search of gems). All four for the C-64; *Crystal Castles* for the VIC as well.

List price for Atarisoft disk games is $34.95; cartridge games, $44.95.

Atari Inc., 1265 Borregas Avenue, P.O. Box 427, Sunnyvale, CA 94086 (phone: 408-745-5752).

Moving to the grimmer side of gaming, here are several releases in a martial mode:

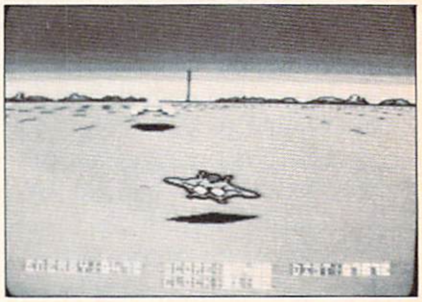
The object of your *Raid over Moscow* is to knock out the Russians' missile launch sites en route to the capital, where you must make a memory of the Soviet Defense Center. On disk or tape for the C-64; $39.95.

Access Software, 925 East 900 South, Salt Lake City, UT 84105 (phone: 801-532-1134).

As MicroProse's *Nato Commander* you must fend off a Soviet Bloc invasion of Europe. The scrolling real-time simulation arms you with air, land, sea, and—as a last resort—nuclear forces. $34.95 on disk or cassette for the C-64.

MicroProse Software, 10616 Beaver Dam Road, Hunt Valley, MD 21030 (phone: 301-667-1151).

A combat quartet from SSI: *50 Mission Crush* sends you on 50 bombing raids in a B-17 Flying Fortress. C-64 disk; $39.95.



Topple the Dark Tower with **Stealth.**
READER SERVICE NO. 230

You must build a different type of *Fortress* in the game so named, then use it to ward off enemy attacks while fighting to gain control of the surrounding countryside. On disk for the C-64; price is $34.95.

The first two C-64 releases in the *When Superpowers Collide* series are *Germany 1985* (diskette, $59.95) and *RDF 1985* (diskette, $34.95). In the former you direct NATO forces against a Soviet invasion of West Germany; in the latter, you marshal the United States' Rapid Deployment Force against a red strike against the Persian Gulf. Both include maps, data cards, and rule book.

Strategic Simulations Inc., 883 Stierlin Road, Bldg. A-200, Mountain View, CA 94043-1983 (phone: 415-964-1353).

The first of Adventure International's Marvel Superhero games, mentioned here in April, is available. It stars the *Hulk* and will be distributed by Commodore for the C-64 and Plus/4. (Commodore has also released the arcade adaptations *Satan's Hollow* and *Solar*

Fox.)

Commodore Business Machines, 1200 Wilson Drive, West Chester, PA 19380 (phone: 215-431-9100).

From Broderbund for the C-64: *Raid on Bungeling Bay* (thought we'd left the battlefield behind, didn't you?) picks up where *Choplifer* left off—with the gamer in a helicraft, charged with the destruction of The War Machine.

If the 150 levels of *Lode Runner* weren't enough for you, *Championship Lode Runner* provides 50 more of increased complexity.

Castles of Dr. Creep strives to re-create the mood of the old-time horror movies in sending the player through 13 castles with a total of over 200 rooms.

Spelunker throws ghosts, bats, and natural obstructions against the gamer traveling through a series of underground passageways in search of treasure.

Whistler's Brother sends a pair of characters through jungles, mountains, lava beds, and the like in search of misplaced tools and documents.

You'll require great *Stealth* to cross an artillery-dotted landscape and destroy the Dark Tower.

All on disk, $29.95 each (except *Championship Lode Runner*—$34.95).

Broderbund Software, 17 Paul Drive, San Rafael, CA 94903 (phone: 415-479-1170).

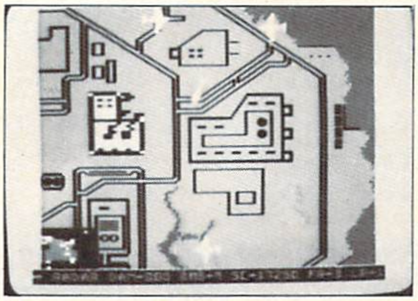
GET FILTHY

Keep missing the lottery? Make your fortune the old-fashioned way with *Get Rich: Strategies*, first in a series of programs from Arrays, Inc./Continental Software designed to teach money management skills.

The package consists of *Worksheets*, enabling the user to set financial goals; *Calculations*, for solving problems involving money, time, and interest; and *Graphs*, for analysis of data over a period of time. For the C-64; $49.95.

Arrays, Inc./Continental Software, 11223 South Hindry Ave., Los Angeles, CA 90045 (phone: 213-410-3977).

If the few measly million you make with the aforementioned program doesn't satisfy you, there's also *Financial Analyst* from Excelsior. Five sub-programs help the 64 user manage his finances; a sixth sub-program lets him review financial options. $34.95, disk or cassette.



Make a **Raid on Bungeling Bay.**
READER SERVICE NO. 231

Excelsior Software Company, 516 Fifth Avenue, New York, NY 10036 (phone: 212-398-9748).

SLOT MACHINE

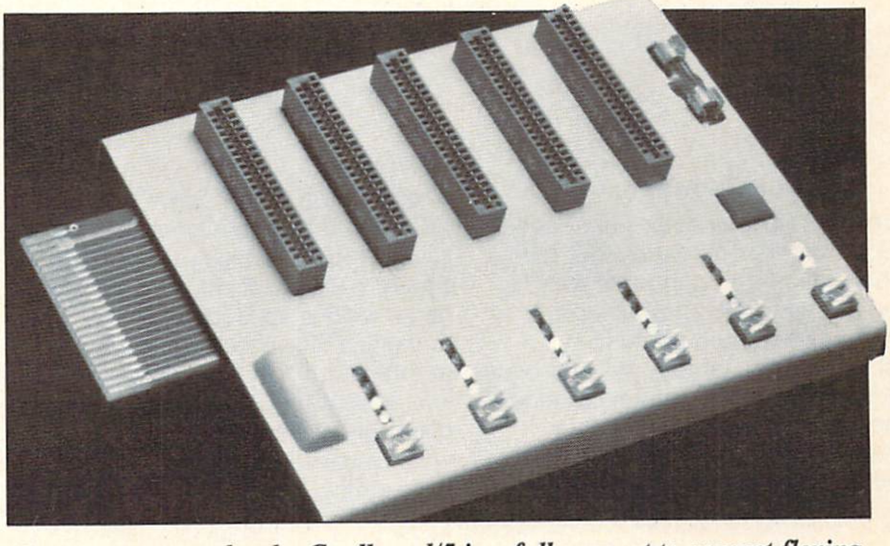
The Cardboard/5 from Cardco provides an alternative to the tedium of switching cartridges. Five slots are provided, each with four LED's to indicate its status and two toggle switches to control power for each cartridge and cartridge request honoring. The system allows a user to supply power to a cartridge without causing an auto-start. $79.95; for the C-64.

Cardco, Inc., 313 Mathewson, Wichita, KS 67214 (phone: 316-267-3807).

GO ALL THE WAY

Once you've got your brain as high-teched as it will go, how can you tech up even higher? The means is at your fingertips. . .with the Dvorak keyboard, offering 30%-80% increased speed over the standard QWERTY keyboard. *Q-Vert* by Q.A.D. Systems is a program that will convert the 64 or VIC (+8K) keyboard to Dvorak via software, then take the user through a series of drills. Price is $29.95 plus $2 postage and handling (Ohio residents add 5.5% sales tax).

Q.A.D. Systems, 342 East Schrock Road, Westerville, OH 43081 (phone: 614-460-7433).



Not to worry—under the Cardboard/5 is a full support to prevent flexing.
READER SERVICE NO. 225

LAWN JOB

IMPORTANT: Before typing in this program, refer to pages 67 and 68.

Lawn Job is a game in which you play the role of a lawn boy. As the game begins, you are standing on the walkway in front of your customer's house looking over the task at hand. You must first start your mower by pressing the fire button on the joystick plugged into Control Port 2. As with the typical mower, it will usually take several attempts to start. (Note: mower will not move prior to starting.) After it starts you can begin to mow the lawn.

As you mow, you must be cautious not to hit the tree stumps or large rocks which are scattered about the yard. Hitting these will cause your mower to stop as well as cause excessive wear and tear to your mower. You will want to avoid this as much as possible because the extra time spent restarting the mower plus the added wear and tear can lessen your final earnings considerably.

When your job is finished, you must maneuver your mower to the customer's front porch and press the fire button. This stops your mower and prompts your customer to come out and examine your work. You then must wait patiently while your work is scrutinized. If your work is not finished, you will be told so and you must restart the mower and finish the job.

When the job is completed and the customer is satisfied, your total earnings from the job will be calculated. The amount of time you took (gas money) plus fifty cents for each rock or tree stump that you hit (wear and tear) will be subtracted from your base score in order to determine the total amount of money that you earned for the job.

If you would rather not type in this program and don't want to spring for our monthly disk (see page 66), send a blank disk (1541 format) or a cassette tape, a self-addressed stamped mailer, and $3 to:

Michael Buhidar Jr.
4317 Hermosa
Corpus Christi, TX 78411


```
10 REM LAWN JOB - MIKE BUHIDAR JR
.
16 GOTO11000
17 DIMP(15):P(1)=-40:P(2)=40:P(4)
=-1:P(8)=1
20 RN=0:SC=0:CO=54272:HC=0:GOSUB2
000:GOSUB200:TI$="000000":GOSUB30
0
120 BO=128:BC=13:PE=PEEK(56320):J
V=15-(PEAND15)
125 IFP=5THENSC=SC+.050001:SC$=ST
R$(SC):SC$=MID$(SC$,2,4)
127 PRINTTAB(10);SC$;TAB(32);MID$
(TI$,3,2)":";RIGHT$(TI$,2)"{CU}"
130 PP=OP+P(JV):DN=PEEK(PP):P=PEE
K(PP+CO)AND15:IFDN<>128THEN400
137 IFJV=0ORJV=5ORJV=6ORJV=9ORJV=
10THEN120
140 POKEOP,IO:POKEOP+CO,IC:POKEPP
+CO,1:POKEPP,JV+130:OP=PP
145 IO=BO:IC=BC
150 GOTO120
200 REM SOUND INIT.
203 OP=1843:POKEOP,131:POKEOP+CO,
1:IO=32:IC=1
210 S=54272:FORA=STOS+24:POKEA,0:
NEXT:POKES+24,15:POKES+5,63:POKES
+6,255
220 POKES+4,65:POKES+3,10:POKES+2
,255:RETURN
300 REM START MOWER
305 PRINT"{WH}"TAB(32);MID$(TI$,3
,2)":"RIGHT$(TI$,2):PRINT"{CU}{CU
}"
310 JV=PEEK(56320):FR=(JVAND16):F
ORIT=0TO250:NEXT:IFFR=16THEN305
315 YN=INT(RND(1)*6)+1:IFYN=1THEN
340
320 FORTT=2TO0STEP-1:POKES+1,TT:F
ORBB=255TO0STEP-5:POKES,BB:POKES,
0:NEXT:NEXT
```

FOR THE C-64

by Michael Buhidar Jr.

```
•330 POKES+1,0:GOTO305
•340 FORCC=1TO3:POKES+1,CC:FORHH=0
 TO100:NEXT:NEXT:RETURN
•400 REM COLLISION
•410 IFDN=130THEN500
•420 IFDN=129THEN520
•430 IFDN=133AND(PEAND16)<>16THEN5
 40
•450 GOTO120
•500 REM HIT ROCK
•505 FORBC=15TO0STEP-1:POKE53280,B
 C:NEXT
•510 POKEOP,JV+130:FORB=4TO0STEP-.
 05:POKES+1,B:NEXT:RN=RN+1:GOSUB30
 0:GOTO120
•520 REM WALK ON SIDEWALK
•530 BO=129:BC=15:GOTO137
•540 REM CHECK FOR COMPLETE JOB
•545 LL=0:PRINT"{CU}{CU}{CU}
    LOOKING OVER YOUR WORK."
•546 FORGG=1TO0:POKES+1,GG:FORYY=2
 55TO0STEP-1:POKES,YY:NEXT:NEXT:PO
 KES+1,0
•550 FORV=1024TO2023:PV=PEEK(V+CO)
 AND15:GOSUB560:NEXT
•552 IFLL=MMTHEN600
•555 PRINT"{CU}       YOU HAVE NOT F
 INISHED THE JOB!"
•556 FORG=0TO3000:NEXT:PRINT"{CU}

      {CD}{CD}"
•558 GOSUB300:GOTO120
•560 IFPV=5THENLL=LL+1
•570 RETURN
•600 REM END OF GAME
•605 FS=VAL(SC$)-VAL(TI$)/100-.50*
 RN:IFFS>HSTHENHS=FS
•610 POKES+1,0:POKE53281,0:PRINT"{
 SC}{WH}{CD}{CD}{CD}{CD}{CD}{CD}{C
 D}{CD}{CD}{CD}{CD}"
•615 IFFS<0THEN700
•620 PRINTTAB(6)"GREAT JOB, YOU EA
 RNED $";FS
•625 PRINT:PRINTTAB(10)"HIGHEST EA
 RNING $";HS
•630 PRINT"{CD}{CD}{CD}{CD}{CD}{CD
 }{CD}{CD}{CY}";:PRINTTAB(8)"PRESS
  TRIGGER TO PLAY AGAIN"
•640 AA=PEEK(56320)AND16:IFAA=0THE
 N20
•650 GOTO640
•700 PRINT"{CU} YOUR EXPENSES OUTW
 EIGHED YOUR EARNING!"
•710 PRINTTAB(5)"FIND ANOTHER WAY
 TO EARN MONEY!":GOTO630
•2000 REM INITIALAZATION OF SCREEN
•2003 POKE53281,0:POKE53280,0:POKE
 53265,PEEK(53265)AND239
•2005 PRINT"{SC}{WH}{CD}{CD}{CD}{C
 D}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD
 }{CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD}
 {CD}{CD}":PRINTTAB(2)"SCORE:  0"T
 AB(25)"TIME:  ";
•2010 POKE53281,5:POKE53280,0:POKE
 53281,9
•2015 FORB5=1464TO1783:POKEB5,128:
 POKEB5+CO,5:NEXT
•2016 FORC2=1247TO1447STEP40:POKEC
 2,128:POKEC2+CO,5:NEXT
•2017 FORC3=1248TO1448STEP40:POKEC
 3,128:POKEC3+CO,5:NEXT
•2018 FORC4=1249TO1449STEP40:POKEC
 4,128:POKEC4+CO,5:NEXT
•2019 FORC5=1250TO1450STEP40:POKEC
 5,128:POKEC5+CO,5:NEXT
•2020 FORC6=1310TO1326:POKEC6,128:
 POKEC6+CO,5:NEXT
•2021 FORC7=1350TO1366:POKEC7,128:
 POKEC7+CO,5:NEXT
•2022 FORC8=1390TO1406:POKEC8,128:
 POKEC8+CO,5:NEXT
•2023 FORC9=1430TO1446:POKEC9,128:
 POKEC9+CO,5:NEXT
•2029 FORA1=1864TO1903:POKEA1,69:P
 OKEA1+CO,0:NEXT
•2030 FORA2=1784TO1823:POKEA2,129:
 POKEA2+CO,15:NEXT
•2040 FORA3=1323TO1801STEP40:POKEA
 3,129:POKEA3+CO,15:NEXT
•2050 FORA4=1324TO1328:POKEA4,129:
 POKEA4+CO,15:NEXT
•2060 POKE1288,129:POKE1288+CO,15:
 FORZ1=1247TO1250:POKEZ1,133:POKEZ
 1+CO,12:NEXT
•2070 FORA5=1028TO1148STEP40:POKEA
 5,116:POKEA5+CO,HC:NEXT
•2080 POKE1188,76:POKE1188+CO,HC
•2090 FORA6=1189TO1210:POKEA6,111:
 POKEA6+CO,HC:NEXT
•3000 FORA7=1251TO1451STEP40:POKEA
 7,116:POKEA7+CO,HC:NEXT
•3010 FORA8=1211TO1055STEP-39:POKE
 A8,78:POKEA8+CO,HC:NEXT
•3020 FORA9=1492TO1499:POKEA9,111:
 POKEA9+CO,HC:NEXT
•3030 POKE1491,76:POKE1491+CO,HC:P
 OKE1496,76:POKE1496+CO,HC
•3040 FORB1=1056TO1456STEP40:POKEB
 1,116:POKEB1+CO,HC:NEXT
•3050 POKE1500,122:POKE1500+CO,HC
•3060 FORB2=1060TO1460STEP40:POKEB
 2,106:POKEB2+CO,HC:NEXT
•3070 FORB3=1271TO1285:POKEB3,67:P
 OKEB3+CO,12:NEXT
•3080 POKE1270,74:POKE1270+CO,12:P
 OKE1286,75:POKE1286+CO,12
```

```
•3090 POKE1230,93:POKE1230+CO,12:P
OKE1246,93:POKE1246+CO,12
•3100 FORB4=1231TO1245STEP2:POKEB4
,135:POKEB4+CO,7:NEXT
•3110 FORB6=1024TO1424STEP40:POKEB
6,128:POKEB6+CO,5:NEXT
•3120 FORB7=1025TO1425STEP40:POKEB
7,128:POKEB7+CO,5:NEXT
•3130 FORB8=1026TO1426STEP40:POKEB
8,128:POKEB8+CO,5:NEXT
•3140 FORB9=1027TO1427STEP40:POKEB
9,128:POKEB9+CO,5:NEXT
•3150 FORB9=1228TO1428STEP40:POKEB
9,128:POKEB9+CO,5:NEXT
•3160 FORC1=1229TO1429STEP40:POKEC
1,128:POKEC1+CO,5:NEXT
•3170 FORD1=1061TO1461STEP40:POKED
1,128:POKED1+CO,5:NEXT
•3180 FORD2=1062TO1462STEP40:POKED
2,128:POKED2+CO,5:NEXT
•3190 FORD3=1063TO1463STEP40:POKED
3,128:POKED3+CO,5:NEXT
•3195 FORE1=1024TO1744STEP40:POKEE
1,72:POKEE1+CO,1:NEXT
•3197 FORE2=1063TO1783STEP40:POKEE
2,71:POKEE2+CO,1:NEXT
•3199 MM=INT(RND(1)*10)+10
•3200 FORTT=1TOMM
•3210 RP=INT(RND(1)*1000)+1024
•3220 IFPEEK(RP)<>128THEN3210
•3221 IFPEEK(RP-1)=130THEN3210
•3222 IFPEEK(RP+1)=130THEN3210
•3223 IFPEEK(RP+39)=130THEN3210
•3224 IFPEEK(RP+41)=130THEN3210
•3225 IFPEEK(RP-41)=130THEN3210
•3226 IFPEEK(RP-39)=130THEN3210
•3230 POKERP,130:NEXTTT
•4000 POKE53265,PEEK(53265)OR16
•9999 RETURN
•11000 REM REDEFINE CHARACTERS
•11001 IFPEEK(12288)=60THEN11045
•11004 POKE53280,0:POKE53281,0:PRI
NT"{SC}"
•11005 PRINTTAB(16)"{LG}{CD}{CD}{C
D}{CD}{CD}{CD}{CD}{CD}LAWN JOB":P
RINTTAB(19)"{WH}{CD}BY"
•11006 PRINTTAB(12)"{CD}MIKE BUHID
AR JR."
•11007 PRINT"{CY}{CD}{CD}{CD}{CD}{
CD}{CD}{CD}{CD}REDEFINING CHARACT
ERS, PLEASE WAIT..."
•11010 PRINTCHR$(142):POKE52,48:PO
KE56,48:CLR:POKE56334,PEEK(56334)
AND254
•11020 POKE1,PEEK(1)AND251
•11025 FORCH=0TO1023:POKECH+12288,
PEEK(CH+53248):NEXT
•11030 FORCD=0TO87:READD:POKECD+13
312,D:NEXT
•11040 POKE1,PEEK(1)OR4:POKE56334,
PEEK(56334)OR1
•11045 POKE53272,(PEEK(53272)AND24
0)+12
•11050 GOTO17
•12000 REM DATA FOR CHARACTERS
•12010 DATA173,255,219,254,183,253
,111,255
•12020 DATA255,255,255,255,255,255
,255,255
•12030 DATA255,239,231,227,195,193
,126,255
•12040 DATA255,231,231,189,36,126,
102,60
•12050 DATA60,102,126,36,189,231,2
31,255
•12060 DATA255,255,255,255,255,255
,255,255
•12070 DATA240,230,255,149,149,255
,230,240
•12080 DATA0,28,54,28,8,42,28,0
•12090 DATA0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0
•12110 DATA15,103,255,169,169,255,
103,15
```

BUG REPELLENT LINE CODES FOR LAWN JOB

```
LINE # 10:AP
LINE # 16:IP
LINE # 17:GE
LINE # 20:MP
LINE # 120:NN
LINE # 125:IK
LINE # 127:FD
LINE # 130:DD
LINE # 137:HA
LINE # 140:HO
LINE # 145:OJ
LINE # 150:CD
LINE # 200:ID
LINE # 203:EG
LINE # 210:BL
LINE # 220:IN
LINE # 300:NM
LINE # 305:PF
LINE # 310:PC
LINE # 315:KB
LINE # 320:MF
LINE # 330:JH
LINE # 340:CB
LINE # 400:FH
LINE # 410:KA
LINE # 420:KO
LINE # 430:EF
LINE # 450:CD
LINE # 500:KI
LINE # 505:NC
LINE # 510:GC
LINE # 520:NA
LINE # 530:BM
LINE # 540:DN
LINE # 545:KK
LINE # 546:JI
LINE # 550:CP
LINE # 552:LM
LINE # 555:LN
LINE # 556:GA
LINE # 558:IK
LINE # 560:JE
LINE # 570:IM
LINE # 600:CD
LINE # 605:OP
LINE # 610:FM
LINE # 615:EL
LINE # 620:CE
LINE # 625:DB
LINE # 630:MF
LINE # 640:FK
LINE # 650:CG
LINE # 700:EM
LINE # 710:PD
LINE # 2000:PH
LINE # 2003:HE
LINE # 2005:JG
LINE # 2010:FB
LINE # 2015:PB
LINE # 2016:HG
```

```
LINE # 2017:PB    LINE # 3040:LN    LINE # 3200:EN    LINE # 11020:IM
LINE # 2018:MM    LINE # 3050:HE    LINE # 3210:EC    LINE # 11025:BF
LINE # 2019:DL    LINE # 3060:CP    LINE # 3220:OD    LINE # 11030:LG
LINE # 2020:OI    LINE # 3070:JE    LINE # 3221:NN    LINE # 11040:IE
LINE # 2021:LD    LINE # 3080:IH    LINE # 3222:NM    LINE # 11045:LI
LINE # 2022:JB    LINE # 3090:FG    LINE # 3223:AB    LINE # 11050:PA
LINE # 2023:ML    LINE # 3100:DJ    LINE # 3224:CI    LINE # 12000:FO
LINE # 2029:LD    LINE # 3110:AF    LINE # 3225:CP    LINE # 12010:PD
LINE # 2030:MG    LINE # 3120:HA    LINE # 3226:FA    LINE # 12020:MG
LINE # 2040:KM    LINE # 3130:ML    LINE # 3230:GK    LINE # 12030:AI
LINE # 2050:HA    LINE # 3140:IG    LINE # 4000:AN    LINE # 12040:NG
LINE # 2060:LJ    LINE # 3150:LK    LINE # 9999:IM    LINE # 12050:IM
LINE # 2070:KF    LINE # 3160:NB    LINE # 11000:OI   LINE # 12060:MG
LINE # 2080:KL    LINE # 3170:EI    LINE # 11001:JD   LINE # 12070:AI
LINE # 2090:PD    LINE # 3180:KD    LINE # 11004:AD   LINE # 12080:FE
LINE # 3000:FM    LINE # 3190:CG    LINE # 11005:PC   LINE # 12090:FG
LINE # 3010:JH    LINE # 3195:CH    LINE # 11006:EI   LINE # 12110:LF
LINE # 3020:FE    LINE # 3197:GJ    LINE # 11007:IB   LINES: 134
LINE # 3030:PG    LINE # 3199:KC    LINE # 11010:JH
```

FLOTSAM

I have found that a LOAD"$$",8 followed by a LIST command will give you a listing of the DISK NAME, ID and the free space available. Yes, the $$ is correct. I have not seen this command in any publication or documentation about the Commodore 64, and believe it to be of great value when only the DISK NAME, ID and available space are important.

—Kent R. Wagner
Cincinnati, OH

As a reader of *Ahoy!*, I have enjoyed your programs as well as the articles and reviews. My main difficulty has been in reading graphic program lines without a clear idea about how many spaces are between the symbols. In my hacking, I was forced to learn some tricks to overcome this problem.

My latest effort was my most rewarding. Upon typing in the game title for *Post Time* (June), and finding it illegible, I decided to take matters into my own hands. First, I utilized the same PRINTTAB statements, but changed the colors and symbols in the quote mode to suit myself. I ran each line to see how it appeared on the screen and eventually got it all up there in a satisfying fashion. Since this involved only 10 or so lines, it wasn't long before it was completed. I then cursored through the listing and identified the spaces between the symbols. I then wrote out my listing on a piece of paper and used lower case (sp) for one space and (5sp) for 5 spaces, etc. I then erased the title lines from the completed program and inserted my lines in about 10 minutes!

Thanks for the learning experience. Keep up the good work!

—H. Gene Harless
Eugene, OR

Many readers have complained of similar problems typing in our programs, even in recent issues when we've printed the number of spaces in a line when there are many. In our next issue, we'll be introducing a new program listing format that will identify any character that occurs more than three times. This will include all characters including spaces and graphics characters. Additionally, graphics characters will be identified in a more understandable manner. Come next issue, typing in Ahoy! *programs will be easier than ever!*

After having purchased one copy of your magazine, I would suggest that you spend your efforts in some other area of employment.

—Kent E. Gunnison
Pasco, WA

We cannot reply personally to every letter we receive—and we can of course print only a select few—but we read every letter you write. Your comments help us immeasurably in preparing the type of magazine you want to read. Please continue writing to Flotsam, *c/o* Ahoy!, *45 West 34th Street—Suite 407, New York, NY 10001.*

AN OVERVIEW OF BIT MAPPED GRAPHICS ON THE COMMODORE 64

BY MORTON KEVELSON

While working with the graphics programs reviewed in this issue, I found that an understanding of how Commodore 64 graphics worked was extremely helpful. This was particularly relevant with regard to the manipulation of color with each package. An understanding of just what goes on behind the screen was a great help in figuring out just why some of these programs behaved as they did. The following discussion is intended as an aid in using the graphics packages available for the Commodore 64.

BIT MAPPING

The term bit mapping refers to a one to one correspondence between a single bit in the computer's memory and a single pixel on the video display. The organization of the Commodore 64 bit map is strongly related to the structure of character graphics. As with character graphics, the display of the Commodore 64 is organized as 40 columns by 25 rows of individual characters or "cells." Each of these characters is stored as eight bytes of memory, where the individual bits of each byte make up the shape of the character. Since there is room for 1,000 characters on the screen, it becomes apparent that 8,000 bytes will be required to uniquely bit map the entire screen.

The Commodore 64 has two possible bit mapped modes. These will be referred to as high resolution bit mapped mode or hi-res mode and multicolor bit mapped mode. The two modes are fundamentally similar, differing primarily in the way each mode handles the display of color. High resolution mode allows up to 320 horizontal dots to be plotted on the screen. Multicolor mode only displays 160 dots. Both modes display 200 dots in the vertical direction.

LOCATING THE BIT MAP

Before describing the mechanisms of bit mapping, let us see just where it is located in memory. The specific hardware of the Commodore 64 is the determining factor. On top of it all is the VIC II chip itself. This large scale integrated circuit, which for all practical purposes is a self-contained graphics computer, is designed to "see" only sixteen kilobytes of memory at one time. Given the 64 kilobyte address range of the 6510 microprocessor, there are four possible locations for the VIC II chip's addressable memory. Each of these 16 kilobyte sections will be referred to as a "bank."

The particular bank seen by the VIC II is controlled by bits zero and one of Port A on the CIA #2 chip. This port is located at memory location 56576 ($DD00). Associated with Port A is the data direction register (DDR) located at 56576 ($DD02). The corresponding bits of this location must be turned on to set Port A for output. This is the normal operating state of the Commodore 64. The following statement will ensure the proper setting of the DDR:

```
POKE 56578,(PEEK(56578) OR 3)
```

The actual bank starting address is set by the value of A in the following statement.

```
POKE 56576,(PEEK(56576) AND 25
2) OR A
```

Use the following table to select the proper value of A.

BANK	A	START ADDR	VIC CHIP RANGE
0	3	0	$0000-$5FFF
1	2	16384	$6000-$7FFF
2	1	32768	$8000-$BFFF
3	0	49152	$C000-$FFFF

All of the graphic data must be located in the same bank if it is to be accessible to the VIC II chip. Bank 0 is the power up bank for the Commodore 64.

HI-RES MODE

Figure 1 shows how the memory locations in the bit map relate to their position on the display screen. The first byte (numbered 0) corresponds to the eight horizontal pixels in the upper left corner of the display. The second byte accounts for the row of pixels directly beneath and so on through the eighth byte. The ninth byte is back up on the top row. This pattern repeats till the 40th cell position, in the upper right corner, which starts with the 313th byte. A total of 320 bytes make up the first row of 40 cells. This pattern repeats for each of the 25 rows of cells. The second row starts with byte number 320, the lower left corner of the display corresponds to byte number 7,687, and the lower right corner belongs to byte number 7,999.

FIGURE 1
BYTE ARRANGEMENT OF THE BIT MAP

ROW #	COLUMN # 1	2 · · · · · ·	· · · 40
	0	8	312
	1	9	313
	2	10	314
1	3	11	315
	4	12	316
	5	13	317
	6	14	318
	7	15	319
	320		:
	321		:
	322		:
	323		:
2	324		:
	325		:
	326		:
	327		:
	:		:
	7,680		7,992
	7,681		7,993
	7,682		7,994
25	7,683		7,995
	7,684		7,996
	7,685		7,997
	7,686		7,998
	7,687		7,999

To turn on the top left pixel of the screen, you would POKE a 128 into the first byte. This is because the bit which corresponds to this position is the leftmost or most significant bit. Table 1 gives the values of each of the bit positions. Mathematically speaking, the value of each bit position is equal to two raised to the power of that position. Thus, for the first position, two to the zero is one. For the second position, two to the one is equal to two and so on till the last position where two to the seventh power is equal to 128. To turn on the first and fourth pixels, you would POKE 128 + 16 = 144 into the first byte. The maximum value of a single byte is of course 255.

TABLE 1 — BIT VALUES

Bit Position	7	6	5	4	3	2	1	0
Value	128	64	32	16	8	4	2	1

The 8,000 byte bit map can have only two possible locations relative to the start of the 16K video bank. This is controlled by bit 3 of the VIC-II chip Memory Control Register at memory location 53272 ($D018). If this bit is off, the bit map starts at the first byte of the video bank. If bit 3 is on, the bit map shifts up 8,192 bytes. Bit 3 can be controlled by:

```
POKE 53272,(PEEK(53272) AND 247)
OR B
```

If B=0 the offset will be zero; if B=8 the offset will be 8,192.

Remember, to find the actual address of a bit map location, you will have to add three numbers. The first is the bit map byte address from Figure 1. The second is the bank starting address from the table. The third is the location of the bit map in the bank, either 0 or 8192. For example, the starting address for the bit map in the upper half of bank 2 would be 0 + 16384 + 8192 = 24576.

Hi-res graphics mode is turned on by setting bit 5 of the VIC-II control register at location 53265 ($D011) to a 1:

```
POKE 53265,PEEK(53265) OR 32
```

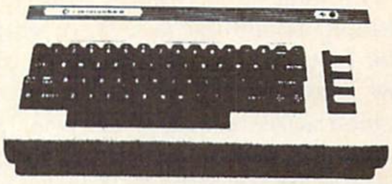
COLOR AND HI-RES

There is one more thing which must be taken care of to display a pixel, and that is the display color. In hi-res mode, the Commodore 64 can display two colors in each character cell out of the sixteen possible colors. Furthermore, each of the 1,000 cells can independently contain any combination of two out of sixteen colors. Thus, you can see that a hi-res screen can be rather colorful.

The information which tells the VIC II chip what colors to display is stored in screen memory. This is a 1,000 byte area located at 1024 ($0400) on power up. The actual location of screen memory can be set by the user to any 1K section of the block by setting the upper four bits of memory location 53272 ($D018) to the proper value:

```
POKE 53272,(PEEK(53272) AND 15) O
R C
```

commodore 64



COMMODORE 64 HARDWARE

1530 Datasette	65.00
Maxiset	45.00
1541 Disk Drive	239.00
1650 Modem	89.00
1526 Printer	269.00
1702 Color Monitor	239.00
MPS 801 Printer	219.00

PRINTERS

GEMINI 10X (80 Column)	279.00
GEMINI 15X (136 Column)	419.00
DELTA 10 (80 Column)	419.00
RADIX 10 (80 Column)	639.00
RADIX 15 (136 Column)	749.00
POWERTYPE Daisywheel	379.00
EPSON RX-80 (80 column)	339.00
EPSON RX-80 FT (80 column)	469.00
EPSON FX-80 (80 Column)	555.00
EPSON FX-100 (136 Column)	749.00
SILVER-REED EXP 500 Daisywheel (80 column)	399.00
OKIDATA 92P	449.00

PRINTER RIBBONS

Gemini Printers (Black/Blue/Red/Purple)	3.00
Epson Printers	7.49

COMMODORE INTERFACE CABLES

Cardco GT	79.00
Tymac	79.00

MODEMS

HES Modem I	54.00
Transterm Software	29.00

ACCESSORIES

Data 20 (80 CHR)	129.00
Data 20 (80 CHR/CPM)	229.00
Koala Pad	69.00
Numerical Keyboard	35.00
Light Pen	22.00
Cardco Exp. Int.	55.00

DISK DRIVE

Concourde	309.00

MONITORS

Gorilla™ Hi Res 12" Non-Glare Screen	89.00
Gorilla™ Hi Res 12" Non-Glare Amber Screen	99.00
Sakata SC 100 Color Screen	239.00
NEC 1260	109.95
Monitor Cable	10.00
Sakata Monitor Stand	15.00

GENERIC DISKS

Generic 100% Defect-Free/Guaranteed MINI-FLOPPY DISKS

Diskettes (1 Box Min.) - 10 per box

	SS/DD	DD/DD
1 or 2 Boxes	17.49/box	20.99/box
3 - 9 Boxes	15.99/box	19.99/box
10+ Boxes	14.99/box	18.99/box

Bulk Diskettes with Sleeves - Price per Disk

	SS/DD	DD/DD
10-29	1.59	1.99
30-99	1.49	1.89
100+	1.45	1.79

TO ORDER CALL **TOLL FREE**

1-800-824-7506

(513) 294-2002 (To order in Ohio)

COMPUTER CREATIONS, INC.

P.O. Box 292467 - Dayton, Ohio 45429

For information call: Tues/Thurs. 11 a.m. to 3 p.m. Eastern Standard Time.

MasterCard - Visa - C.O.D. (Add $2.50). All orders add $3.50 shipping and handling in Continental United States. Actual freight will be charged outside U.S. to include Canada, Alaska, Hawaii, Puerto Rico and A.P.O's.

Reader Service No. 227

"The Rabbit"

for your VIC 20 or CBM 64 — $39.95



If you own a VIC 20 or a CBM 64 and have been concerned about the high cost of a disk to store your programs on . . . worry yourself no longer. Now there's the RABBIT. The RABBIT comes in a cartridge, and at a much, much lower price than the average disk. And speed . . . this is one fast RABBIT. With the RABBIT you can load and store on your CBM datasette an 8K program in almost 30 seconds, compared to the current 3 minutes of a VIC 20 or CBM 64, almost as fast as the 1541 disk drive.

The RABBIT is easy to install, allows one to Append Basic Programs, works with or without Expansion Memory, and provides two data file modes. The RABBIT is not only fast but reliable. (The Rabbit for the VIC 20 contains an expansion connector so you can simultaneously use your memory board, etc.)

MAE — NOW THE BEST FOR LESS!

Please for your own protection consider the MAE first before you buy that other assembler. We've had numerous customers who wasted their money on some cheaper off brand assembler tell us how much better the MAE is.

The most powerful Macro Assembler/Editor available for the Commodore 64 and other CBM/PET computers, and also for the ATARI 800/XL and Apple II/IIE.

MAE includes an Assembler, Editor, Word Processor, Relocating Loader, and more all for just $59.95.

We could go on and describe the MAE but we thought you would like to read our customers' comments. The following are actual unedited comments from correspondence about the MAE:

"Excellent Development Package."
"Compares to DEC and INTEL."
"My Compliments to Carl Moser and EHS."
"It is a superb program."

TELSTAR 64 — "A Star is Born"

Sophisticated Terminal Communications Cartridge for the 64.

PFO 10D 00D CP D1 D2 BELL 12:30:00 10:14:36
(TELSTAR's Status Line)

Don't settle for less than the best!

- Upload/Download to/from disk or tape. Menu-driven.
- Automatic File Translation. Real Time Clock plus Alarm Clock.
- Communicates in Industry Standard ASCII.
- Line editing capability allows correcting and resending long command lines.
- 9 Quick Read functions.
- Similar to our famous STCP Terminal package.
- Works with Commodore Modems and supports auto-dialing.

The best feature is the price — **only $49.95** (Cartridge and Manual)

Machine Language Monitor Cartridge

for the CBM 64

More than 20 commands allow you to access the CBM 64's Microprocessors Registers and Memory Contents. Commands include assemble, disassemble, registers, memory, transfer, compare, plus many more.

Someday every CBM 64 owner will need a monitor such as this.

Cartridge and Manual — **$24.95**

Eastern House

3239 Linda Dr.
Winston-Salem, N.C. 27106
(919) 748-8446
Send for free catalog!

VISA · MasterCard

Reader Service No. 228

The value of C is taken from Table 2. Screen memory is laid out in a fairly straightforward fashion. The first byte corresponds to the character cell in row 1 column 1. The values proceed in an orderly fashion across each row till byte 1,000 corresponds to row 25 column 40. Figure 2 shows the screen memory layout. As with the bit map, you will have to add the bank start address (Table 1) and the screen offset (Table 2) to the byte value from figure 2 to determine the actual address of screen memory.

FIGURE 2
BYTE ARRANGEMENT OF SCREEN MEMORY

ROW # \ COLUMN #	1	2	3	39	40
1	0	1	2	38	39
2	40	41	42	78	79
3	80	81	82	118	119
:	:				:
25	960	961	962	998	999

In character mode, screen memory tells the VIC-II which character to display. In hi-res mode, each location of screen memory controls the background and foreground color of the corresponding bit map cell.

TABLE 2
SCREEN MEMORY OFFSET FROM BANK START ADDRESS

Offset	C Value	Bit Values of 53272
0	0	0000 XXXX
1024	16	0001 XXXX
2048	32	0010 XXXX
3072	48	0011 XXXX
4096	64	0100 XXXX
5120	80	0101 XXXX
6144	96	0110 XXXX
7168	112	0111 XXXX
8192	128	1000 XXXX
9216	144	1001 XXXX
10240	160	1010 XXXX
11264	176	1011 XXXX
12288	192	1100 XXXX
13312	208	1101 XXXX
14336	224	1110 XXXX
15360	240	1111 XXXX

The information for two colors can be stored in a single byte since only four bits are needed for the 16 possible colors. The first four bits, or the lower nybble, controls the background color. The upper nybble controls the foreground color. A hi-res drawing program such as *DOODLE!* sets the upper nybble with the working paint color. The lower nybble is set by the working paper color. This is the reason why all the pixels in a cell will change color when the working paint color in *DOODLE!* is changed.

MULTICOLOR BIT MAPPED GRAPHICS

The bit map procedure is somewhat more complex in multicolor mode. Pixels are set by a pair of bits rather than a single bit. Each multicolor pixel is twice as wide as a hi-res pixel. Thus, a single byte controls only four pixels rather than eight. As a result, a horizontal line displays only 160 pixels instead of 320. The tradeoff results in greater color flexibility. A bit of binary math reveals that a pair of bits can take on four possible values. Thus, a multicolor pixel can be one of four possible colors. Each character cell in multicolor mode can display up to four different colors.

As in hi-res mode, two of the colors are stored as nybbles in the 1,000 byte screen memory. The third color is stored in the 1,000 byte color memory starting at memory location 55296 ($D800). The location of color memory is fixed in the Commodore 64. Its layout is the same as for screen memory (Figure 2). This allows these three colors to be uniquely determined, for each character cell, in multicolor mode. I will refer to the color, which is determined by color memory, as the primary color. The colors set by the low and high nybbles of screen memory will be referred to as auxiliary colors 1 and 2 respectively. This designation is somewhat arbitrary since I did not find an official Commodore designation for them. The fourth color is common for the entire bit mapped display. It is controlled by the contents of memory location 53281 ($D021). Changing this location will change the background color of the entire screen. Table 3 summarizes the multicolor bit assignments.

TABLE 3 — MULTICOLOR BIT ASSIGNMENTS

Bits	Value	Color Info Stored At
00	0	Background 53281
01	1	Upper Nybble Screen Memory
10	2	Lower Nybble Screen Memory
11	3	Color Memory Nybble

Multicolor mode is controlled by bit 5 at location 53265 ($D011) and bit 4 at location 53270 ($D016). Two statements are required to set multicolor mode:

```
POKE 53265,(PEEK(53265) OR 32

POKE 53270,(PEEK(53270) OR 16
```

The concepts of bit mapped graphics may seem very complex at first. They are actually not all that difficult. The explanation presented here is intended as an aid for a better understanding of how the various drawing programs perform. The actual manipulation of bit mapped graphics involves large numbers of repetitive calculations. It is possible to write the necessary routines in BASIC, but the result will be rather slow. If you wish to seriously pursue the subject, the books listed below will be helpful in getting you started.

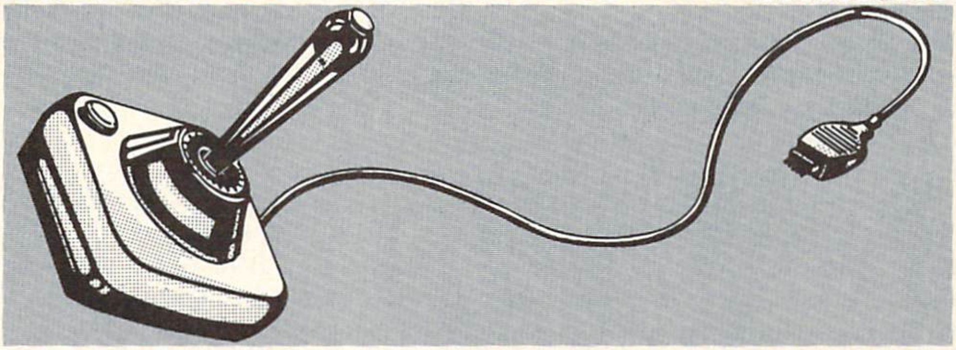
To summarize, the on/off information for a pixel is stored in the bit map. Color information for hi-res

Continued on page 90

RUPERT REPORT

JOYSTICK!

COMMUNICATING WITH



REMOTE MECHANICAL INTERFACE DEVICES

Jo Ann Case

By Dale Rupert

It's fun to use joysticks. This month we'll investigate ways of using joysticks in BASIC programs on the Commodore 64. If you don't have a joystick for your computer, you might read along anyway, since in the process, we will also look at string arrays and ways of improving program execution speeds.

There are two nine-pin jacks on the side of the Commodore 64 labeled Control Port 1 and Control Port 2. One or two switch-type (digital) joysticks with nine-pin plugs may be connected to the control ports. We will not discuss potentiometer-type joysticks, sometimes called paddles, which may also be used on the C-64. The joysticks used with most videogame machines are compatible with the C-64 and are the type discussed here.

How does the joystick talk to the computer once it is plugged in, and what does it say? The *C-64 User's Guide* that came with the computer is not any help. On page 141, it shows the names of the pins on the two jacks, but that's all. Fortunately the *C-64 Programmer's Reference Guide* provides plenty of information. Beginning on page 343 is a discussion of joysticks, including a BASIC program and a machine language program for using them. (The *VIC 20 Programmer's Reference Guide* provides a good discussion of joysticks for VIC owners.)

The joystick consists of four switches plus a fifth switch called the "fire button." We'll ignore the fire button for the time being. When you push the stick forward, one switch is closed. Pull it backward, left, or right, and different switches are closed, one at a time. The switches are numbered 0 through 3 in the following configuration:

```
       0

2 ---+--- 3

       1
```

Push the stick to the right, and switch 3 is closed, while the other three are open. If you push diagonally forward and to the left, both switches 0 and 2 are closed. Pulling diagonally backward and to the right closes switches 1 and 3, and so forth. There are eight positions with either one or two switches closed. The ninth position is with the joystick centered, and none of the switches are closed.

Since the joystick is such an obviously digital device, it should not be surprising that the computer communicates very easily with it. According to the *Programmer's Reference Guide,* information from the joysticks is stored in locations 56320 and 56321.

Plug in a joystick, or two if you have them, and try the following program. I suggest you label your joysticks with the number of the port that each is plugged into.

```
10 PRINT PEEK(56320);PEEK(56321)
20 GOTO 10
```

When you run this program, the computer tells you what it sees in the two memory locations mentioned above. The value in 56320 is shown in the left-hand column, and the value on the right is from memory location 56321. With both joysticks in the center position, the screen shows columns with 127 on the left and 255 on the right.

Move the joystick plugged into Control Port 1. The value in the right-hand column changes. Evidently Control Port 1 comes into memory location

56321. By deduction, or by experimentation, you should conclude that Control Port 2 may be read at location 56320. (From the *Programmer's Reference Guide,* I would have thought that Port 1 data is stored at 56320 and Port 2 data is at 56321, but that is obviously incorrect.)

Notice that the values seem to have some pattern to them as you move the stick around. You might also notice that the program slows down whenever the joystick plugged into Port 1 (called joystick 1 from now on) is moved to the left. Do you know which key on the keyboard produces the same effect? Try holding the CTRL key down while this program is running. Surprised? Not only does the CTRL key produce the same effect, it also puts the same value into 56321 as moving joystick 1 to the left. This is our first indication that the joysticks are really nothing more than an extension to the keyboard.

COMPUTER'S INNARDS

Have you ever looked at the schematic diagram of your computer at the back of the *Programmer's Reference Guide*? Have a look at what is inside the computer. Near the upper left of the front side of the diagram you will find the two control ports with the same labels that we saw in the *User's Guide.* We are not going into a course on reading schematic diagrams, but you should be able to see that the wires (lines) from the joystick inputs snake their way down and connect to the same wires that come from the keyboard connector CN1. All of these lines go to the CIA chip, U1.

The computer continuously scans these lines. When a keyboard key is pressed, one of the ROW lines is joined to one of the COL lines. The computer knows which keyswitch is closed by seeing which ROW and COL lines are tied together. The computer treats the joystick switches a little differently, but the joysticks can still be misinterpreted as keyboard information.

For example, stop the program. Move joystick 1 around. You wouldn't know by looking at the screen that those characters weren't generated by the keyboard. Pushing joystick 1 to the left causes the cursor to scoot across the screen, something that the CTRL key doesn't do. So evidently the joystick doesn't exactly duplicate the CTRL key, but if you press the BLK key or the WHT key while holding joystick 1 to the left, you produce the same results as if you had pressed <CTRL>—<BLK> or <CTRL>—<WHT>.

The following program presents the data from the joysticks in a more readable form.

```
10 J1=255-PEEK(56321)
20 J2=127-PEEK(56320)
30 PRINT J1,J2
40 GOTO 10
```

Run the program and move the joysticks around. You will quickly discover that the nine positions of each joystick produce these values:

```
5    1    9
     :
4 -- 0 -- 8
     :
6    2    10
```

The value of J1 or J2 is 1 when the joystick is pushed forward and 4 when it is pushed to the left. Push it forward and to the left to get a value of 5. The value at a diagonal is the sum of the values on either side of the diagonal. We should be able to add a few lines to the program to utilize this information. For example,

```
30 IF J1=1 THEN PRINT "STICK #1 I
S FORWARD"
31 IF J2=6 THEN PRINT "STICK #2 I
S BACK-LEFT"
32 IF J1<>0 AND J2<>0 THEN PRINT
"STOP PUSHING US AROUND!"
```

Once the excitement of these examples has worn off, let's look at ways of speeding up our program's response to the joystick inputs.

QUICKER BASIC

In order to find more efficient ways of communicating with the joysticks, you might try running some time-trials such as the following programs.

```
1 PRINT"TIME TRIAL #1"
5 T0=TI:FOR N=1 TO 1000
10 J1=255-PEEK(56321)
20 J2=127-PEEK(56320)
25 NEXT:T1=TI
30 PRINT T1-T0
```

```
1 PRINT"TIME TRIAL #2"
2 K1=255:K2=127:M1=56321:M2=56320
5 T0=TI:FOR N=1 TO 1000
10 J1=K1-PEEK(M1)
20 J2=K2-PEEK(M2)
25 NEXT:T1=TI
30 PRINT T1-T0
```

The difference between the two programs is that *Time Trial #1* uses numeric constants in lines 10 and 20, and *Time Trial #2* uses numeric variables. Believe it or not, the second program is over twice as fast as the first! On my computer, the first program

took 1533 jiffies (sixtieths of a second) to execute lines 10 and 20 one thousand times, and the second one took only 652 jiffies.

The moral is this: use numeric variables rather than constants for faster programs. This makes sense if you consider that each time the computer sees "56320", it must read each digit and then convert the whole thing to a value in the computer's internal format. On the other hand, when the computer sees "M2", all it has to do is to find the memory locations where it has stored the value of M2.

One thing to be careful of is that you don't define the variables on the inside of the FOR-NEXT loop. Define them only once at the start of your program.

You might be interested to note that combining lines 10 and 20 into a single line saved only 4 jiffies. In fact, combining the NEXT statement with lines 10 and 20 into a single program line saved a total of only 7 jiffies over *Time Trial #2* above. Keep these results in mind when you are tempted to cram seventeen statements into one program line for the sake of speed. The reduced readability and debugability of such a program may outweigh the meager time savings provided.

On some versions of BASIC, variables which are defined first in a program are more quickly found during execution than those defined later. You might investigate this for yourself to see if that is true on your computer. If so, you should define the most-frequently used or most time-critical variables ahead of the others, even if they aren't used until later.

Spaces within a program line make a small difference in speed. A space after J1 in line 10 above takes 4 jiffies longer than the same line with no space. That's four-sixtieths of a second for the execution of one thousand spaces, or roughly 67 microseconds (yes, 67 millionths of a second!) per space. Once again weigh the readability of your program with extra spaces against the time savings without them.

USING ARRAYS

A handy way to utilize the joystick input value is to let it be the index of an array. Because of the funny interaction of the joystick in Port 1 with the keyboard, we will use joystick 2 from now on. Try this program.

```
10 K2=127:M2=56320
20 C$(0)=""
30 C$(1)="HELLO":C$(2)="GOODBYE"
40 C$(4)="LEFT":C$(8)="RIGHT"
50 J2=K2-PEEK(M2)
60 PRINT C$(J2);
70 GOTO 50
```

Lines 20 through 40 define a string array C$. The zeroth element of C$ is the null string. Elements 1, 2, 4, and 8 are given string values as shown. Elements 3 and 5 through 10 are undefined and are therefore also null strings. Line 50 is the same statement we have used before. Recall that J2 will have a value between 0 and 10, depending upon the position of the joystick. In line 60, the present value of J2 determines which element of C$ is to be printed.

For example, if joystick 2 is pushed to the right, the value of J2 as read in line 50 will be 8. Line 60 prints C$(8) which has the value "RIGHT". If you push the stick diagonally right and forward, J2 will equal 9. Since C$(9) equals the null string, nothing will be printed. You may find that either C$(1) or C$(8) is momentarily printed if the joystick briefly goes through either of those positions on its way to the diagonal.

You will see that "HELLO's" are printed very rapidly when the joystick is pushed forward. In some applications you might want this repeating characteristic, such as for moving a cursor around. In other programs, you might prefer to have the chosen value of C$ printed only once upon reaching any joystick position.

You may "debounce" the joystick by adding these two lines:

```
55 IF J2=J0 THEN 50
65 J0=J2
```

Now the program will loop between lines 50 and 55 until the joystick is moved. Once a value of C$ is printed in line 60, J0 is set equal to the current value of J2 in line 65. Again the program will loop between lines 50 and 55 until the value of J2 changes. When it does, the new value of C$ will be printed once and only once. If you change line 20 to

```
20 C$(0) = "*"
```

you will be able to see that C$(0) is in fact printed whenever the joystick returns to the center position. Whether you want debounced or repetitive action on the joystick depends upon its use within your program. For using the joystick to respond to questions or to write out messages, the debounced version is useful. If you want to use the joystick to move an object, such as a cursor, on the screen, the repetitive action is better.

QUICK STICK

The following program shows how you can use the joystick to control the cursor on the screen.

```
5 PRINT CHR$(147)
10 K2=127:M2=56320
```

```
20 LL$=CHR$(157):RR$=CHR$(29)
30 UU$=CHR$(145):DD$=CHR$(17)
40 C$(0)=""
50 C$(1)=UU$:C$(2)=DD$
60 C$(4)=LL$:C$(8)=RR$
70 C$(5)=UU$+LL$:C$(6)=DD$+LL$
80 C$(9)=UU$+RR$:C$(10)=DD$+RR$
90 J2=K2-PEEK(M2)
100 PRINT C$(J2)"*"LL$;
110 GOTO 90
```

Line 5 clears the screen. Line 10 should be familiar to you by now. Lines 20 and 30 define the cursor control variables. LL$ is CHR$(157) which means "Cursor left". (Refer to the *ASCII and CHR$ Codes* appendix at the back of your reference guide.) Since C$(4) is set equal to LL$ in line 60, whenever this program prints C$(4), the only thing that happens is that the cursor moves one space to the left.

Notice that lines 70 and 80 define the diagonal joystick characters to be combinations of cursor movements. C$(10) moves the cursor down and to the right whenever it is printed. Lines 90 through 110 form the heart of the program. J2 is given the current joystick value in line 90. Line 100 prints the selected joystick character, which in this program is a cursor movement. It then prints an asterisk at the new cursor position, and then it prints one more cursor-left character.

Why the last LL$ in line 100? Without it, we can't draw vertical lines. After the asterisk is printed, the computer automatically moves the cursor one space to the right. The LL$ simply brings the cursor back to the asterisk. Try it without the LL$ and see what happens.

When you run this program, you will notice some things about it that you may not like. For one, the motion is very fast, and therefore, not easily controllable. Secondly, it scrolls off the screen and wraps around to the other side, or it causes the whole screen to scroll up. Thirdly, watching an asterisk move around on the screen tends to be less than interesting after an hour or two (maybe less). And finally, you adventurous ones have found out that pressing the fire button doesn't shoot down any aliens. Instead it shoots down the program which doesn't like it a bit.

FINALE

Now that you understand the workings of the previous program, let's fix up its problems and turn it into a utility that might be useful in your own program. The following program is a start.

```
1 REM--JOYSTICK DOODLER--
2 REM--PRESS KEYBOARD KEYS
3 REM--TO CHANGE PRINTED
4 REM--CHARACTER.
5 PRINT CHR$(147)
10 K2=127:M2=56320
20 LL$=CHR$(157):RR$=CHR$(29)
30 UU$=CHR$(145):DD$=CHR$(17)
35 :REM--CH$=PRINT CHARACTER--
40 CH$=CHR$(169)+LL$
45 :REM--CR$=CURSOR CHARACTER--
50 CR$=CHR$(166)+LL$
55 :REM-C$()=MOVEMENT DIRECTION--
60 C$(0)=""
70 C$(1)=UU$:C$(2)=DD$
80 C$(4)=LL$:C$(8)=RR$
90 C$(5)=UU$+LL$:C$(6)=DD$+LL$
100 C$(9)=UU$+RR$:C$(10)=DD$+RR$
104 :
105 :REM--------MAIN LOOP--------
106 :
110 J2=K2-PEEK(M2)
115 :REM-BUTTON PRESSED, CLR SCRN
120 IF J2=16 THEN PRINT CHR$(147)
:GOTO110
125 IF J2>10 THEN 110
130 GET IN$:IF IN$="" THEN 150
135 :REM--NEW CHAR. SELECTED-----
140 CH$=IN$+LL$
150 PRINT CR$;:FOR P=1 TO 10:NEXT
160 PRINT CH$;:FOR P=1 TO 10:NEXT
165 :REM--------PAUSE-----------
170 FOR P=1 TO 20:NEXT
175 :REM-MOVE IN CHOSEN DIRECTION
180 PRINT C$(J2);
185 :REM--FIND CURSOR ROW--------
190 IFPEEK(214)>23 THENPRINT UU$;
195 :REM--FIND CURSOR COLUMN-----
200 IFPEEK(211)>38 THENPRINT LL$;
210 GOTO 110
```

Line 40 defines the string which will be printed at the cursor. Line 50 defines the cursor character. The other lines through 110 perform the functions described in the previous program. Line 120 checks for the fire button. Whenever it is pressed, the value of J2 is increased by 16. If the joystick is centered, the value of J2 is 16. This program only looks for that condition. When the fire button is pressed and the joystick is centered, the screen is cleared. You can modify the program if you want to detect the fire button when the joystick is in any position. If the joystick is pulled back when the fire button is pressed, the value of J2 is 18 (2+16).

Continued on page 90

COMMODARES

PROGRAMMING CHALLENGES

By Dale Rupert

Each month, we'll present several challenges designed to toggle the bits in your cerebral random access memory. We invite you to send your solutions to:

Commodares, c/o **Ahoy!**
P.O. Box 723
Bethel, CT 06801

We will print and discuss the cleverest, simplest, shortest, most interesting and/or unusual solutions. Be sure to identify the *name* and *number* of the problems you are solving. Also show sample runs if possible, where appropriate.

Your original programming problems would be equally welcome!

PROBLEM #10-1: NUMERIC PALINDROME

This problem was suggested by Doug Rider and Dave Steen (McComb, OH). Step 1: the user enters a number. Step 2: the computer adds the reverse of the number to the number. Step 3: If the result is a palindrome (the same digits from right to left as from left to right) or if the result is too large for the computer to handle, the program stops and prints the number of steps required and the palindrome if there is one. Otherwise the computer repeats the whole process from Step 2 with the new result.

Doug and Dave said that this is called the "196 Problem" since even very large computers haven't found the palindrome result for 196. What can you discover? (Send your results as well as your programs.)

PROBLEM #10-2: ROMAN NUMERALS

The user enters a decimal number. The computer prints the Roman Numerals for that number. (The result for 9 must be IX, not VIIII, for example. 1984 is MCMLXXXIV.)

PROBLEM #10-3: UP/DOWN TIMERS

The user enters a number. The computer counts that number of seconds and displays two numbers as it counts. The number on the left is labeled "DOWN" and the number on the right is labeled "UP". If the user enters 20, the left number goes from 20 to 0, and the right number goes from 0 to 20. Both numbers should be displayed at fixed locations near the center of the screen.

PROBLEM #10-4: FAST SHUFFLE

Robert Griffiths (Syracuse, NY) suggested writing the fastest subroutine to shuffle a deck of cards. His best solution so far is shown below, and it takes about 7 seconds.

```
1 REM   PROBLEM #10-4
2 REM     FAST SHUFFLE
3 REM     SUBMITTED BY
4 REM     ROBERT GRIFFITHS
10 DIM C(53),D(53):X=52
20 FOR J=1 TO X:D(J)=J:NEXT
30 REM - MAIN PRGM. GOES HERE -
40 T0=TI:GOSUB 80
50 PRINT (TI-T0)/60 "SECONDS"
60 FOR J=1 TO 52:PRINT C(J),:NEXT
70 END
75 REM   --SHUFFLE ROUTINE--
80 FOR J=X TO 1 STEP -1
90 R=INT(J*RND(1)+1):C(J)=D(R)
100 FORK=RTOJ:D(K)=D(K+1):NEXTK,J
110 RETURN
```

Your subroutine will replace lines 80 through 100. Line 20 sets up the deck in numerical order. Lines 40 and 50 time the subroutine. Line 60 is for debugging purposes to show that the subroutine works properly.

The solution to last month's *Problem #9-1: Passed Words* is the following sequence of inputs:

Input #1: MARC
Input #2: MARCMARC
Input #3: MARC
Input #4: MARCMARC
Get #1: R
Get #2: C

No doubt most of you who tried to break into Marc Spooner's password-protected program agree that he has certainly created an effective deterrent. Congratulations if you did figure out the sequence, but Marc probably has an even tougher scheme by now.

Continued on page 97

BIG FOUR

NEW 128K —MEGA BYTE DUAL DISK DRIVE—80 COLUMN

COMPUTER SYSTEM SALE!

HOME • BUSINESS • WORD PROCESSING



LOOK AT ALL YOU GET FOR ONLY ***$895.***

		LIST PRICE
①	B128 COMMODORE 128K 80 COLUMN COMPUTER	$ 995.00
②	8050 DUAL DISK DRIVE (over 1 million bytes)	1795.00
③	4023 - 100 CPS - 80 COLUMN BIDIRECTIONAL PRINTER	499.00
④	12" HI RESOLUTION 80 COLUMN GREEN OR AMBER MONITOR	249.00
•	BOX OF 10 LORAN LIFETIME GUARANTEED DISKS	49.95
•	1100 SHEETS FANFOLD PAPER	19.95
•	ALL CABLES NEEDED FOR INTERFACING	102.05
	TOTAL LIST PRICE	$ 3717.95

$895. SALE PRICE

Printer replacement options (replace the 4023 with the following at these sale prices)

	LIST	SALE
★ Olympia Executive Letter Quality Serial Printer	$ 699.00	$ 399.00
★ Comstar Hi-Speed 160 CPS 15" Serial-Business Printer	$ 779.00	$ 499.00
★ Telecommunications Deluxe Modern Package	$ 199.00	$ 139.00

Plus You Can Order These Business Programs At Sale Prices

	LIST	SALE		LIST	SALE
Professional 80 Column Word Processor	$149.95	$99.00	Payroll	$149.95	$99.00
			Inventory	149.95	99.00
Professional Data Base	149.95	99.00	General Ledger	149.95	99.00
Accounts Receivable	149.95	99.00	Financial Spread Sheet	149.95	99.00
Accounts Payable	149.95	99.00	Program Generator	149.95	99.00

15 DAY FREE TRIAL We give you 15 days to try out this SUPER SYSTEM PACKAGE!! If it doesn't meet your expectations, just send it back to us prepaid and we will refund your purchase price!!
90 DAY IMMEDIATE REPLACEMENT WARRANTY If any of the SUPER SYSTEM PACKAGE equipment or programs fail due to faulty workmanship or material we will replace it IMMEDIATELY at no charge!!

Write or Call For Free Catalog and Spec Sheets!!

Add $50.00 for shipping and handling!!
$100.00 for Canada, Puerto Rico, Hawaii orders.
WE DO NOT EXPORT TO OTHER COUNTRIES.
Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail! Canada orders must be in U.S. dollars. We accept Visa and MasterCard. We ship C.O.D. to U.S. addresses only.

PROTECTO
ENTERPRIZES (WE LOVE OUR CUSTOMERS)
BOX 550, BARRINGTON, ILLINOIS 60010
Phone 312/382-5244 to order

COMMODORE 64

Computer Learning Pad

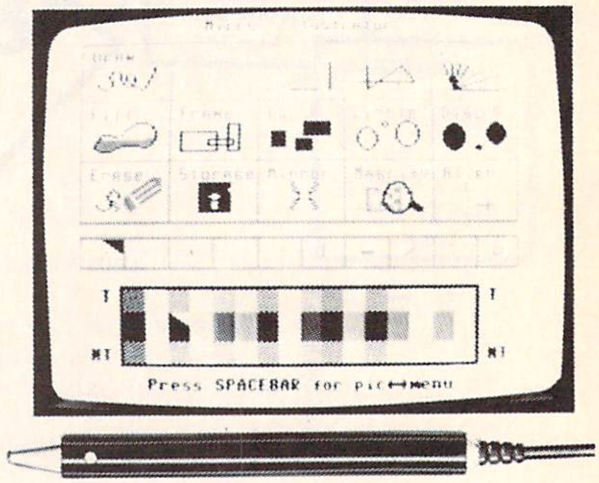
Sale! **39^{95}** List $79.95

- **Makes other graphic tablets Obsolete**
- **Includes Special Printer Feature.**

Now you get this **Fantastic Tech Sketch Computer Learning Pad Program** with a **Professional Light Pen** Free! (39.95 value) plus the **Micro Illustrated Graphics Pad Program** that allows you to draw on your T.V. or monitor screen (better than Gibsons $99 light pen). Whatever you can draw on the screen you can print on your printer (a Tech Sketch Exclusive.)

Hi-Resolution!

NEW FROM TECH SKETCH



JOIN THE "NEW BREED" OF VIDEO ARTISTS!

Light Pens

for your

Commodore 64 & VIC-20

$16.95

Coupon Price $14.95

Economy Model with 3 programs $16.95

Add $3.00 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $6.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES.
Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery. 2 to 7 days for phone orders. 1 day express mail!
VISA — MASTER CARD — C.O.D.
No C.O.D. to Canada, APO-FPO.

PROTECTO
ENTERPRIZES (WE LOVE OUR CUSTOMERS)
BOX 550, BARRINGTON, ILLINOIS 60010
Phone 312/382-5244 to order

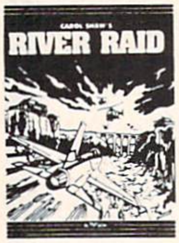
Reader Service No. 241

SUPER SALE!!!

ACTIVISION

GREAT SAVINGS!!!

Fantastic Graphic Arcade Action Games For Your Commodore-64

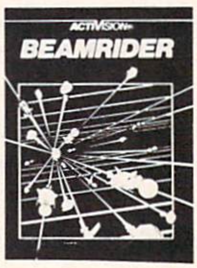


River Raid — Voted Best Action game by Electronic Games Magazine. Now you can play the extended version of the all time great home video game. Over 5000 enemy targets and fantastic action and sound will keep you busy for days. (Highly Recommended) List $39.95. **Sale $27.95**. (Disk)

Pitfall II - Lost Caverns — The feats of Pitfall Harry continue with the full advantages of the Commodore-64. The caverns are 27 levels deep and you must make Harry swim through rivers, float through the air on balloons, even freefall with perfect timing. Fantastic sound, graphics, and action. (Very Highly Recommended) List $39.95. **Sale $27.95**. (Disk)



Beamrider — (Better than Tempest) Move and fire along the grid as aliens attempt to destroy you. Super fast joystick action is the only thing that will keep you alive to make it to the next grid (99 grids) where more challenges await you. Super Action and Graphics. List $39.95. **Sale $27.95**. (Disk)





Decathalon — Finally an Olympic Event for two players at the same time (or play against the computer) Run, Hurl, Vault, Jump and more (10 events in all). Fantastic graphics and sound will keep you playing this one for days on end. Great Competition. List $39.95. **Sale $27.95**. (Disk)

Toy Bizarre — The toy factory has gone crazy. Toys spring to life and try to destroy you and the factory. Jump from level to level and close the valves, and capture the toys before the toys do you and the factory in. Fantastic Graphics and Sound. List $39.95. **Sale $27.95**. (Disk)





H.E.R.O. — Descend through the caverns with your dynamite and laser helmet; clear the tunnels to rescue the trapped miners; watchout for molten lava, animals, insects, etc. If you hit the lights they go out and you're in the dark. Fantastic Graphics and Sound combine with a superb theme to make this a best game for the year. List $39.95. **Sale $27.95**. (Disk)



Zenji — Find your way through the paths. Easy? The paths constantly change as the squares slide around. Use your senses to decide which way is out. If your senses fail, you are trapped and dead! Fantastic action. List $39.95. **Sale $27.95**. (Disk)

Zone Ranger — Better than Asteroids and Omega Race. You must pilot your starship first through asteroid alien territory and dock at your base, then you must make your way through the mazes past hostile aliens. Constantly 4-way scrolling screens make this a must for all game players. Fantastic Graphics and Action. List $39.95. **Sale $27.95**. (Disk)



Add $3.00 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $6.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES.

Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail!

VISA — MASTER CARD — C.O.D.

No C.O.D. to Canada, APO-FPO.

PROTECTO

ENTERPRIZES (WE LOVE OUR CUSTOMERS)

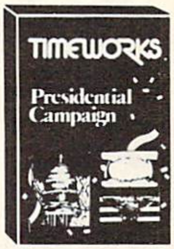
BOX 550, BARRINGTON, ILLINOIS 60010
Phone 312/382-5244 to order

Free Program Offer* TIMEWORKS

Home Management Educational Software

Presidential Campaign

Start with $2,500,000 and a Presidential Nomination. Spend your money wisely to conduct a nationwide campaign to win the confidence of the American Public. At the end of the campaign the American Public will decide on the final decision. Fantastically Realistic. List $29.95. **Sale $19.95**.



**** Buy the Fantastic Presidential Campaign and choose any other Timeworks Program on this page FREE!!***

Tape or Disk

Expires 10/1/84

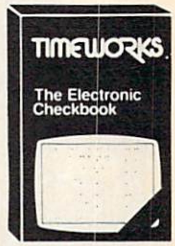
Money Manager



16 categories can be input on a monthly basis for a 12 month period. Full analysis including budget and actual comparisons plus detail by individual items. Colorful bar charts show where you're spending to much or too little. Record all information to tape or disk. List $24.95. **Sale $19.95**.

The Electronic Checkbook

Check recording, sorting and balancing system. Organize and record checkbook transactions for easy recall and sorting by category. A must for tax time. Also produce the sum and average for transaction amounts by indexed categories. Interface with the MONEY MANAGER for graphic detail and actual-budget comparison. List $24.95. **Sale** $19.95.



Your Choice $19.95

Programming Kit I

Learn Basic Programming from the ground up. This first step takes you through the actual design of Lunar Lander, using moving graphics.



Shows in detail how basic programming works and what should go into a good game. List $24.95. **Sale $19.95.**

Your Choice $19.95

Programming Kit II

Intermediate game design and Sprite Builder. Design Slot Machine and study intermediate programming including arrays, subroutines, sound, special function keys, and random function. Plus Sprites are covered in detail and a Sprite Editor is included. List $24.95. **Sale $19.95.**



Programming Kits

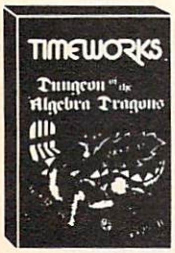
Manuals are written in plain, easy to follow English. The 8 step approach to program design is fully explained and can easily be applied to your future programming adventures.

Programming Kit III



Intermediate Data Base design. Instead of games you may want to go the business route. Use this program to develop a Data Base Manager to allow quick storage and retrieval of any information. Learn disk/tape storage, sorting, menus, plus multi dimensioned arrays, plus a whole lot more. List $24.95. **Sale $19.95.**

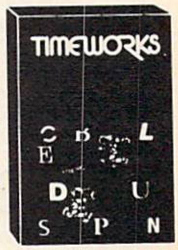
Dungeons Of The Algebra Dragons



At last a good educational game. You must wander through 3-D mazes until you encounter dragons, then you must answer the algebra problems they give you. A right answer gets you a free ride, a wrong answer gets you eaten. Fantastic educational value. (Tape or Disk.) List $29.95. **Sale $24.95**.

Spellbound

The screen is sprinkled with the letters from one of 20 words of your choice along with a few extra random ones. Use your Joystick to sting the letters in the right order but don't sting the wrong ones or cross your path or you explode. Fantastic educational value. Fun for all ages. List $24.95. **Sale $17.95.**



Add $3.00 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $6.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES.

Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery. 2 to 7 days for phone orders. 1 day express mail!

VISA — MASTER CARD — C.O.D.

No C.O.D. to Canada, APO-FPO.

PROTECTO ENTERPRIZES (WE LOVE OUR CUSTOMERS)

BOX 550, BARRINGTON, ILLINOIS 60010

Phone 312/382-5244 to order

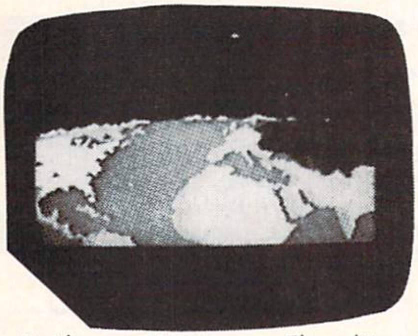
Commodore-64 Software

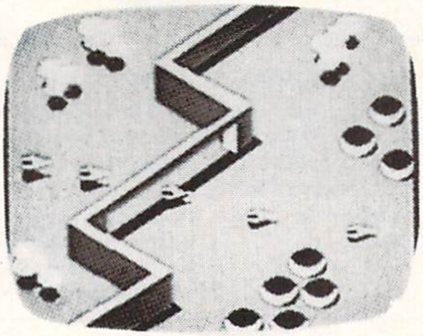
Super Sale!! Save On

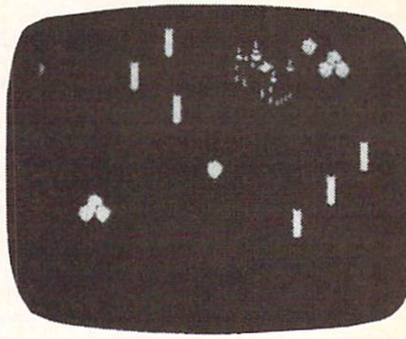
Access Software

Best Multi Level Graphics Games in the U.S.A.

RAID OVER MOSCOW





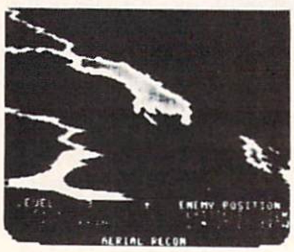


Raid over Moscow — The ultimate in multi screen computer graphics has arrived. (over six screens) The Soviets have launched an attack. First you must locate and destroy the ICBM missiles from your Stealth Bomber, then you must lead your squadron through the Soviet terrain to the Kremlin itself. Destroy the defenses, then destroy the defense center and finally the attack operations themselves. Fly planes, launch grenades, shoot bazookas. Super Fantastic Graphics, Sound, Animation. **(The best multi-level game we've seen!!!)** List $39.95. **Sale $29.95.** (Disk.) *Coupon $26.95.*

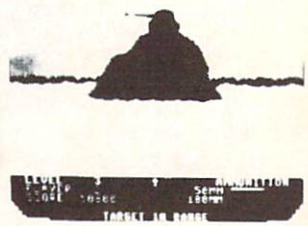
BEACH HEAD

Beach Head — Multi Screen computer graphics at its finest. First you must maneuver your fleet through the mine laden channel while torpedoes are fired at you from all sides. Next you must engage the enemy planes with your anti-aircraft guns (best scenario graphics anywhere). Then you must sink the enemy fleet before they sink you. Next you maneuver to the beach head where your tanks must get through an obstacle course and finally fight the ultimate battle at Headquarters. All scenes in stunning 3-D graphics. (Will be the best graphics of 1984, definitely).

List $39.95. **Sale $24.95.**
Disk or Tape.

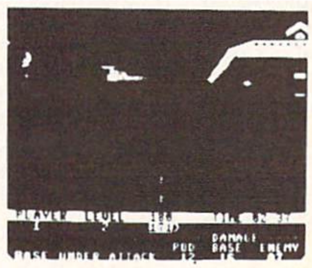






Actual Screens ***Actual Screens***

NEUTRAL ZONE



Neutral Zone — Reach the outer edges of the galaxy. You must protect your spaceship from attacking enemy space ships while defending yourself from enemy missiles. Fantastic 360 degree 3-D High Resolution Graphics. Rated No. 1 for graphics in 1983. List $34.95. **Sale $24.95.**(Tape or disk).

Add $3.00 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $6.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES.

Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail!

VISA — MASTER CARD — C.O.D.

No C.O.D. to Canada, APO-FPO.

PROTECTO

ENTERPRIZES (WE LOVE OUR CUSTOMERS)

BOX 550, BARRINGTON, ILLINOIS 60010
Phone 312/382-5244 to order

SEGA
THE ARCADE WINNERS

Arcade Classics Come To The Commodore 64

STAR TREK
STRATEGIC OPERATIONS SIMULATOR
THE VIDEO GAME

• Official arcade version.



Now you can control the Starship Enterprise as you fight Klingons before they destroy your bases. Full 3-D view as well as overhead radar lets you know what's going on around you. Use your joystick to control warp drive, impulse power, photon torpedoes and phasers. But watch your shields, photon supply and warp power. Fantastic graphics and sound make this a must for everyone. List $39.95. **Sale $29.95** (cartridge).

Congo Bongo

• Official arcade version • 3-D graphics • Two screen displays • One or two players.



The famous arcade game featuring the coconut throwing gorilla, monkeys, hippos, sharks and rhinos now comes to the Commodore 64. Superb reproduction of the arcade machine challenges you to destroy the gorillas lair. List $39.95. **Sale $29.95** (cartridge)

ZAXXON

• Official arcade version • Stunning 3-D scrolling graphics • Multiple screen displays.



Now the famous invasion of robot Zaxxon's lair can be done in your own home. Invade the fortress, get past the fighters then through Zaxxon's home and finally destroy Zaxxon himself. Just like the arcade game. This one's fantastic. List $39.95. **Sale $29.95** (cartridge).

BUCK ROGERS
PLANET OF ZOOM

• Official arcade version • 3-D color graphics • Four Galactic screen displays.



Just like the arcade version. Steer your ship through deadly electron posts as you fight off enemy space saucers and hoppers. Finally you meet the enemy source ship. If you get through all this you start again only this time the enemies have more powers and surprises. Fantastic graphics and sound. List $39.95. **Sale $29.95** (cartridge).

Add $3.00 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $6.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES.
Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail!
VISA — MASTER CARD — C.O.D.
No C.O.D. to Canada, APO-FPO.

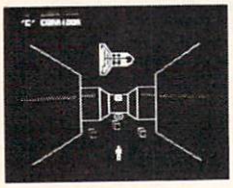
PROTECTO
ENTERPRIZES (WE LOVE OUR CUSTOMERS)
BOX 550, BARRINGTON, ILLINOIS 60010
Phone 312/382-5244 to order

Reader Service No. 245

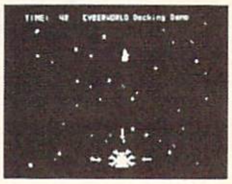
Super Action Software!

Commodore 64 — PROGRESSIVE PERIPHERALS & SOFTWARE — **Disk Programs**

(World's Greatest Adventure Game)



CYBERWORLD
List $49.95 **Sale $34.95**



NEW

This five-screen arcade adventure packs the computer with intense graphics and sound! You are a special Cyberleague agent in a universe full of hostile aliens and vicious robots. Joystick and keyboard transport you through 3-D rooms, space barriers, fleets of invaders, and warship-ridden quadrants of space. Over 100 sprites, 8 new character sets, and dozens of mind-boggling sound effects make up this multi-layered adventure. Animation, action, and strategy all combined into a game so extensive that two disk sides are jammed with game programs and data! Reach the ultimate rank of admiral and you may carve a niche in the permanent high-score list. A full-size book quality manual with full-color covers is included to guide you through your most exciting game experience.

NEW

Perplexian Challenger

List $34.95 **Sale $24.95**

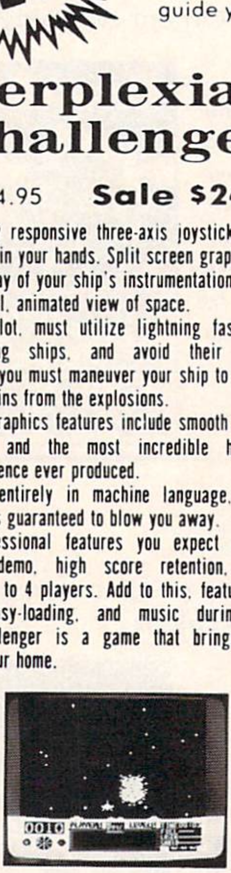
The incredibly responsive three-axis joystick control of a space fighter is in your hands. Split screen graphics provide a continuous display of your ship's instrumentation, as well as a three-dimensional, animated view of space.

You, as a pilot, must utilize lightning fast reflexes to destroy invading ships, and avoid their return fire. Simultaneously, you must maneuver your ship to capture space debris that remains from the explosions.

Outstanding graphics features include smooth 3-D rotations, split screens, and the most incredible high-resolution hyperspace sequence ever produced.

Programmed entirely in machine language, this action-strategy game is guaranteed to blow you away.

All the professional features you expect are included: automatic self-demo, high score retention, pause, and provisions for 1 to 4 players. Add to this, features you don't expect like easy-loading, and music during the load. Perplexian Challenger is a game that brings the arcade experience to your home.



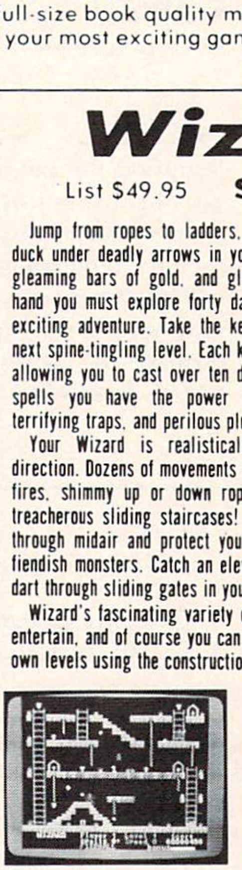
Wizard

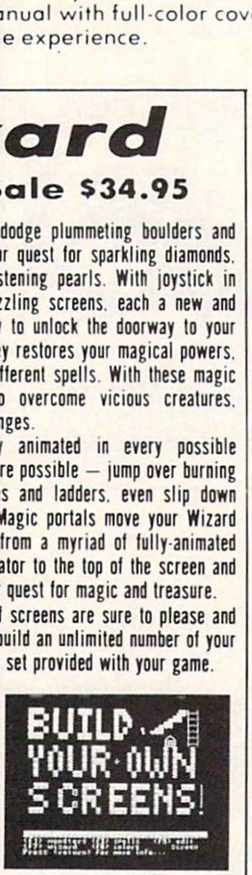
List $49.95 **Sale $34.95**

Jump from ropes to ladders, dodge plummeting boulders and duck under deadly arrows in your quest for sparkling diamonds, gleaming bars of gold, and glistening pearls. With joystick in hand you must explore forty dazzling screens, each a new and exciting adventure. Take the key to unlock the doorway to your next spine-tingling level. Each key restores your magical powers, allowing you to cast over ten different spells. With these magic spells you have the power to overcome vicious creatures, terrifying traps, and perilous plunges.

Your Wizard is realistically animated in every possible direction. Dozens of movements are possible — jump over burning fires, shimmy up or down ropes and ladders, even slip down treacherous sliding staircases! Magic portals move your Wizard through midair and protect you from a myriad of fully-animated fiendish monsters. Catch an elevator to the top of the screen and dart through sliding gates in your quest for magic and treasure.

Wizard's fascinating variety of screens are sure to please and entertain, and of course you can build an unlimited number of your own levels using the construction set provided with your game.





Gothmog's Lair

List $44.95 **Sale $34.95**

Real-time adventure excitement at its best. Solve countless puzzles and slay over a dozen monsters by using the huge vocabulary of over 200 words. Two challenging difficulty levels await you with over 80 areas, each fully described in Old English script.

Menacing monsters, kniving villains, tattered code books and treacherous terrain are just a few of the situations you must overcome in your quest for the thirteen priceless treasures. More than seventy objects are invaluable to you in your search for glory and wealth!

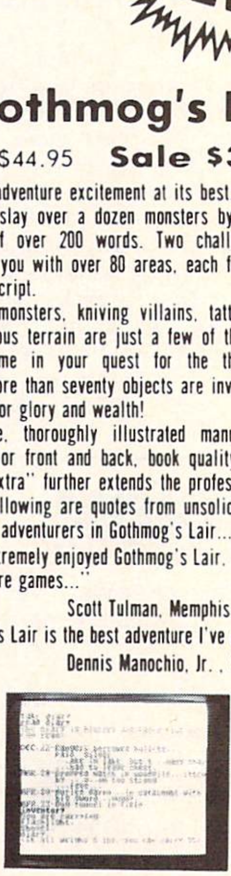
A full-size, thoroughly illustrated manual is included. Featuring color front and back, book quality, and a fold-out map, this "extra" further extends the professionalism of this game. The following are quotes from unsolicited testimonials sent to us by adventurers in Gothmog's Lair...

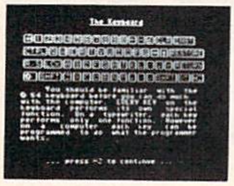
"I have extremely enjoyed Gothmog's Lair, and plan to buy more adventure games..."

Scott Tulman, Memphis, TN

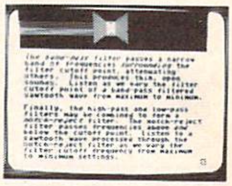
"Gothmog's Lair is the best adventure I've ever played..."

Dennis Manochio, Jr., Saratoga, CA





PROFESSOR
List $39.95 **Sale $29.95**



An in-depth self-tutorial for the Commodore 64 on a two-sided disk. This menu-driven tutorial covers every aspect of your C-64—BASIC, keyboard, sound, music, simple and advanced graphics. Quizzes test your comprehension at the end of many lessons! On-screen illustrations, sound effects and full-color interactive graphics make learning easy and interesting. The PROFESSOR is your ONLY choice for an all-in-one, thorough tutorial about the Commodore 64!

Add $3.00 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $6.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES.

Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery. 2 to 7 days for phone orders, 1 day express mail!

VISA — MASTER CARD — C.O.D.

No C.O.D. to Canada, APO-FPO.

PROTECTO ENTERPRIZES (WE LOVE OUR CUSTOMERS)

BOX 550, BARRINGTON, ILLINOIS 60010
Phone 312/382-5244 to order

Reader Service No. 246

EPYX COMPUTER SOFTWARE
Strategy Games for the Action-Game Player

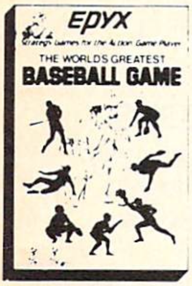
*Buy 1 (at Sale Price!) Get CRASH CRUMBLE CHOMP FREE

Commodore 64

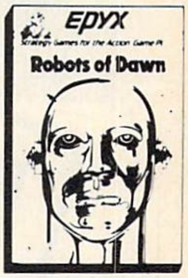
(Expires 12-25-84)

(Limit: One Per Customer) See Below

The Best Arcade Games Around



World's Greatest Baseball — Finally, a great sports game. Realistic right down to the playing field. Contro.l all the actions -even the outfielders. Play realistic baseball all year round. (Fantastic graphics, action, and sound!) (Disk) List $34.95. **Sale $24.95.**



Robots of Dawn — Elijah Baley is here from the best-selling science fiction series by world renowned author Isaac Asimov. Now you can become Earth's most famous detective in this exciting test-adventure. Question the inhabitants of far flung cultures — who's lying, who's telling the truth, and who is trying to murder you? Jump into the action in this fascinating epic murder mystery. Can you piece together the elusive clues to solve the eternal question "who done it?" (Disk) List $39.95 **SALE $29.95**

PROTECTO EXCLUSIVE

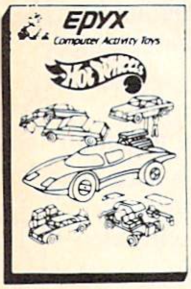


Barbie — The biggest name in dolls. Browse through Barbie's closet full of beautiful clothing and dress her for the party. You can cut her hair or make it longer and color it or change the style. Buy new clothing at the boutique or any of the six other specialty shops, or even go to the dress shop and create Barbie's designer clothes with the help of the computer. The combinations are endless and so is the fun. One or two players. (Disk) List $39.95 **SALE $29.95**



G.I. Joe — The best selling toy soldier is now available as an activity toy on the home computer. Select the battle situation then choose the equipment you think you will need to get the job done. Choose from a number of uniforms and weapons in your well stocked arsenal and get ready for the action. Play alone or with a friend, if you plan the right strategy you will complete the mission if not you will have to try again. One or two players. (Disk) List $39.95 **SALE $29.95**

PROTECTO EXCLUSIVE



Hot Wheels — Now all the action and fun from playing with Hot Wheels cars is taken one step further on the home computer. Hot Wheels lets you choose your play activity from repairing cars to the destruction derby. You can actually play the activities along with a partner on the computer screen, even build and customize your own cars. A whole new way to play Hot Wheels. One or two players. (Disk) List $39.95 **SALE $29.95**



Break Dance — Break Dance is an action game in which your dancer tries to break through a gang of break dancers descending on him. "A simon-like game" where your dancer has to keep the steps of the computer controlled dancer, and even a free dance segment where you develop your own dance routines and the computer plays them back for you to watch. Now anyone can break dance!! (Disk) List $34.95 **SALE $24.95**

***Buy any Epyx game from Protecto and choose Silicon Warrior, Lunar Outpost, or Upper Reaches of Apshai FREE! ($39.95 value)**

In case of stock outage we will substitute.

Add $3.00 for shipping, handling and insurance. Illinois residents please add 6% tax. Add $6.00 for CANADA, PUERTO RICO, HAWAII, ALASKA, APO-FPO orders. Canadian orders must be in U.S. dollars. WE DO NOT EXPORT TO OTHER COUNTRIES.
Enclose Cashiers Check, Money Order or Personal Check. Allow 14 days for delivery, 2 to 7 days for phone orders, 1 day express mail!
VISA — MASTER CARD — C.O.D.
No C.O.D. to Canada, APO-FPO.

PROTECTO
ENTERPRIZES (WE LOVE OUR CUSTOMERS)
BOX 550, BARRINGTON, ILLINOIS 60010
Phone 312/382-5244 to order

Graphics PROCESSING

AND THE COMMODORE 64

BY MORTON KEVELSON

The graphics capabilities of the Commodore 64 are second to none among home computers. This built-in graphics capability is one of the reasons behind the popularity of the Commodore 64. After all, a 40 column by 25 line text display plus 16 colors is nothing to sneer at in a home computer. Add to that a selection of dot graphics modes with a maximum resolution of 320 by 200 dots, and impressive results can be achieved. You've seen some of these results among the many graphically excellent games on the market.

Unfortunately, Commodore has not made it easy for the average user to get at these capabilities. A Commodore 64, fresh out of the box, requires innumerable PEEKs and POKEs to access the high resolution graphics from the built-in BASIC. The results are likely to be rather limited and slow. Machine language is required to get any meaningful speed, but how many of us have the ability or time to write long ML programs? One possibility is to enhance the built-in BASIC with a commercial extension. These programs add powerful graphics commands to Commodore BASIC, allowing the easy programming of graphic designs. This approach is probably the best solution for most graphic programmers.

Many of us would still like to play with the machine's graphics without having to write our own software. Fortunately, a number of enterprising companies have elected to provide for our needs. These programs will turn your Commodore 64 into a video color sketchpad upon which you may indulge in visual composition to your heart's content. They will almost, but not quite, do for graphics what a word processor does for text (see the *Graphics Challenge* on page 4 of this issue).

Before reading the actual reviews, I suggest that you browse through the article on bit mapped graphics on page 15. The material presented there will assist in the appreciation of just what these packages can do. The explanation of how color works will be especially helpful. It turns out that the way each program handles color plays an important role in its overall operation.

PROGRAM FEATURES

Before getting down to the actual reviews, let us take a look at some of the features these packages have in common.

Draw mode—each of these programs allows you to move a pixel around the screen leaving a colored trail. The action is very similar to doodling with a colored crayon. One package has even elected to go by that very name. The color of the crayon can be set by the user.

Brush or cursor—refers to the crayon with which you will draw. All of the packages allow adjustment of the brush size or shape, very often both. Some of the programs offer considerable flexibility in this regard. One even lets you grab a section of the sketch and draw with it.

Text mode—refers to the ability to enter text as part of the graphic image.

Geometric shapes—the programs all have the ability to automatically draw the common geometric shapes. Straight lines, rectangles, triangles, and circles are most popular. Some allow solid shapes as well as outlines to be produced.

Rays—allows you to continuously draw a sequence of lines radiating from a common point.

Paint or fill—allows an area of the screen to be painted with a selected color. Some of the pro-

grams require the area to be completely enclosed while others are more flexible.

Patterns or textures—a very powerful feature which draws or paints in a repeating pattern of pixels rather than a solid line. Impressive effects are surprisingly easy to achieve. Several of the programs come with a predefined set of patterns. Some will let you design your own.

Oops—a very helpful feature which allows you to undo the most recently executed command.

Copy—allows you to define a selected portion of the drawing and repeat it one or more times at other parts of the drawing. Some packages allow you to copy between alternate screens.

Alternate screens—the ability to store a second drawing in the computer's memory.

Merge—the consolidation of two separate drawings, either from an alternate screen or by loading in from disk.

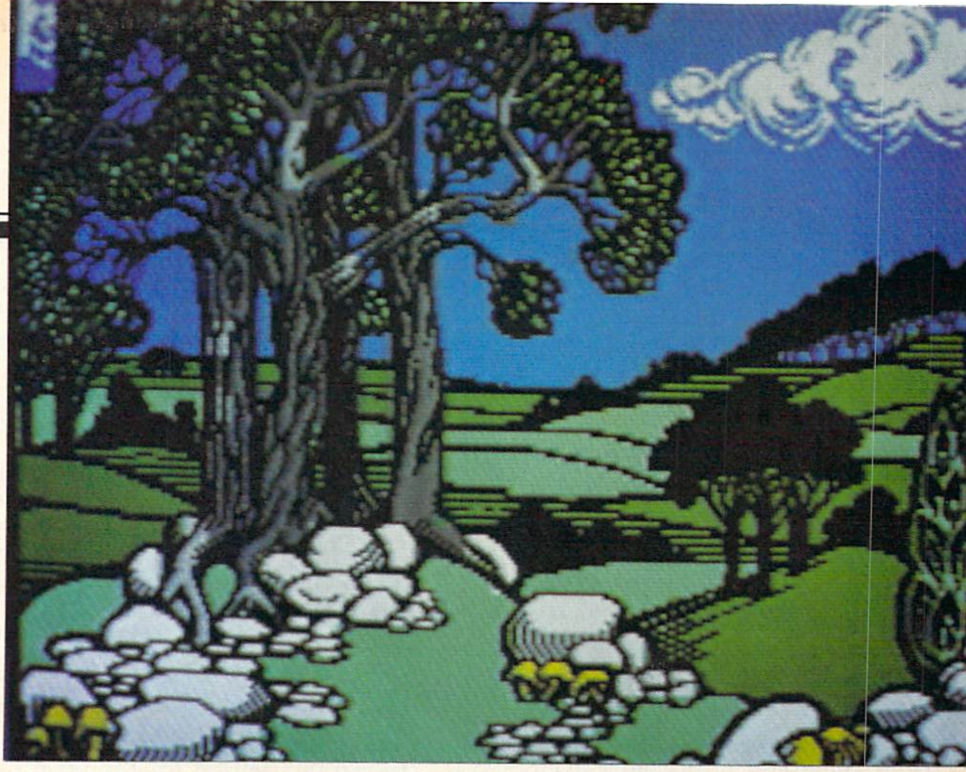
Zoom—allows you to enlarge a small portion of the screen to work on individual pixels. This feature is very important as otherwise the input devices lack sufficient precision to reliably manipulate the image on a pixel basis.

Input Device—the mechanism used to control the pencil on the screen. Just about every conceivable input device showed up among the packages reviewed. Included are the keyboard, joystick, trackball, digitizing pad, light pen, and a mechanical drawing tablet. (Many programs allow a choice of more than one input device.)

Mirror—the ability to create a mirror image of the sketch, either along a horizontal or vertical axis or both.

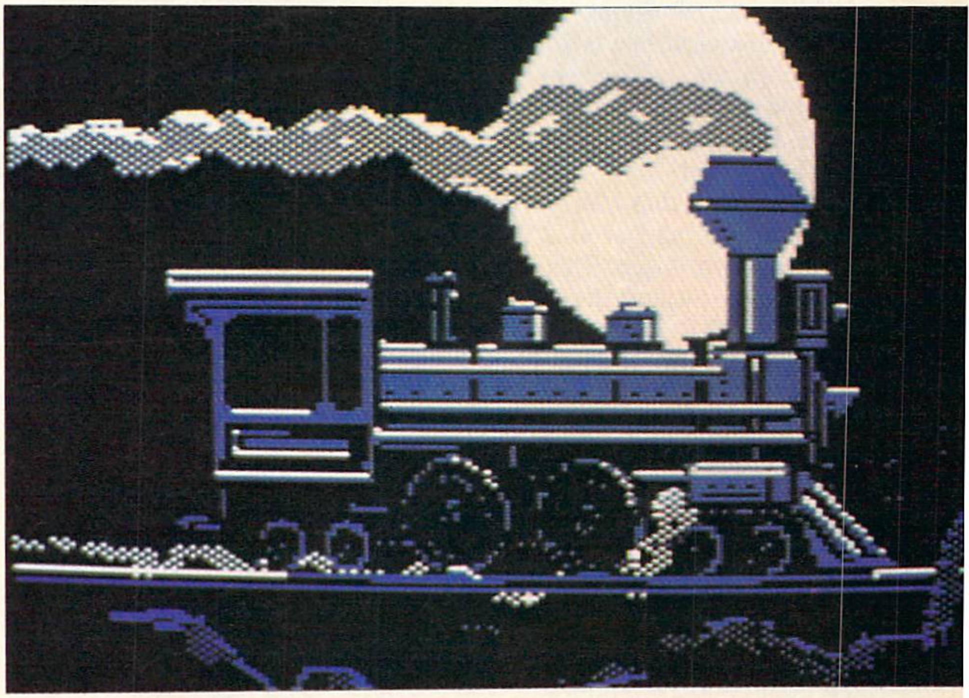
Change color—allows you to instantaneously change all pixels in a given display color to a different color.

Printer output—the ability to make a hard copy of the image on a dot matrix printer is very useful. Only two of the programs reviewed supplied this as a built-in feature and one offers it as an option. There is also a separate program from an independent source



Middle Earth *(drawn on* **DOODLE!***) is by Wayne Schmidt—as are most of the renderings accompanying the graphics articles in this issue and next.*





Trains *by Art Huff is a good example of the graphics possibilities of* **Paint Magic** *(so good an example that the picture is supplied with the disk).*

for just this purpose.

Aspect Ratio—refers to the proportions of height to width of the printed output. If the aspect ratio is too far off, circles will be flattened into ovals and the picture will appear to be flattened or drawn out.

Save and Load—the ability to store images on disk or tape for later recall.

ON LIGHT PENS AS INPUT DEVICES

Two of the packages used a light pen as the input device. This device can actually sense the position of the electron beam on the display monitor as it creates the image. To understand what is involved, consider that the Commodore display is updated 60 times a second. 240 vertical lines make up each frame. The 64 uses only 200 of these for the active display area. The rest make up the border. The length of each line is 480 dots based on the 320 dots used by the 64. Simple math shows that each line takes only 69 microseconds (millionths of a second) to be scanned. The time for each dot is less than two tenths of a microsecond.

The VIC-II chip keeps count of the scan line number as it is creating each image. This number is stored in memory location 53268 ($D014). The chip also keeps track of the electron beam's horizontal position in location 53267 ($D013). When the light pen is activated by the scanning of the electron beam, the contents of these two locations are latched or frozen until they have been read or PEEKed from BASIC. At this time the registers are released until the light pen is activated once again.

As it turns out in the real world, things are never perfect. In this case there are some small but significant delays associated with the light pen and its electronics. These delays have the effect of causing the triggering signal from the light pen to arrive at the VIC chip a short time after the electron beam has illuminated the spot at which the light pen was pointing. The net effect is to have the VIC chip think the pen was pointing to the left of its actual position.

The light pen software should take care of this offset by calibrating the equipment when it is first turned on. This is usually done by displaying a mark at a known screen location and comparing the light pen reading to the actual position. The offset value is then stored for use by the program.

Using a light pen with a television or color monitor can be a bit tricky. For one thing, there is the expected jitter associated with a manually held instrument. Associated with this is a shift from true vertical. The distance of the monitor surface to the phosphor coating may be significant. If the thickness of the CRT glass and the safety screen are taken into account, we may have a distance of a quarter of an inch. The result is that the light pen may read the screen at some location other than where you think you are pointing. The associated software should try to take these variables into account by averaging several consecutive light pen readings. Of course too much averaging tends to slow things down. In spite of its high speed, the electron beam still takes one sixtieth of a second to return to a particular spot.

We did find one peculiarity associated with the light pen. When the system is first turned on, the light pen readings were very erratic for about fifteen minutes. This happened with both light pen and software packages we tested. It seems to be caused by the circuitry of the Commodore 64 itself. As the computer warms up, the light pen-sensing circuits seem to drift. If this should occur with your equipment, we suggest that you let it warm up for fifteen minutes or so before using.

THE REVIEWS

For this report we looked at nine graphics packages and one stand-alone multicolor print utility. Two of these were designed for use with light pens. One uses a touchpad as an input device. Another uses a linked mechanical arm to generate an analog input. The remainder will accept input from a joystick, trackball, or keyboard. You will also find some actual samples of what can be done with these packages (given enough time and talent, of course). Some of these images were manufacturer's samples provided with the programs. A number of them were actually produced for this report by Wayne Schmidt, a member of our local Commodore users group.

You may recognize *Middle Earth* by Mr. Schmidt. A copy is supplied on the *DOODLE!* disk. It is also being used by Okidata in the promotional literature for their new Okimate 10 printer. We would like to acknowledge Mr. Schmidt's efforts on behalf of this project.

This month we present a close look at *Koala Painter, Paint Magic, DOODLE!, Peripheral Vision,* and *Picture Perfect.* The last is the multicolor print utility mentioned above. Next month we will look at *Flexidraw, Flying Colors, Computer Crayons, Supersketch,* and *Sorcerer's Apprentice.* We will also give some details on how these packages store picture files on disk and tips on transferring images between some of these packages. □





GRAPHICS PROGRAMS

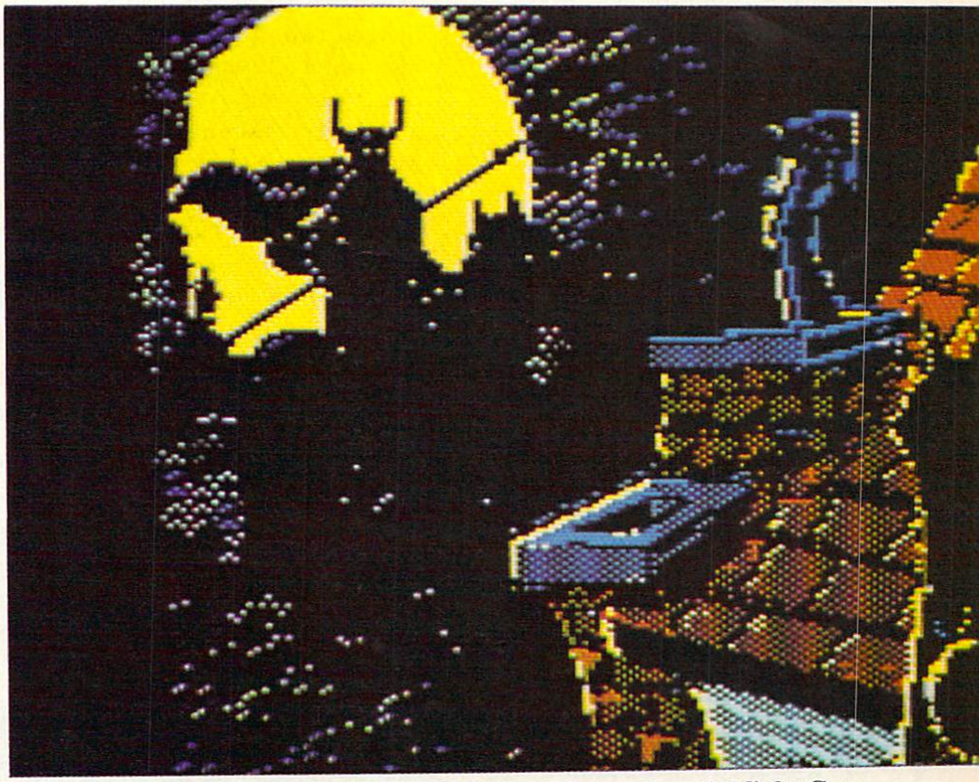
ON THE COMMODORE 64

BY MORTON KEVELSON

Program Name: *Paint Magic*
Type: Multicolor 160 by 200 pixel
Input Device: Joystick, Trackball
Price: $35.00
Datamost Inc.
8943 Fullbright Ave.
Chatsworth, CA 91311-2750
818-709-1202

This excellent program from Datamost has previously been reviewed in *Ahoy!* (August '84, page 49). We are presenting it a second time since this report would not be complete without the inclusion of a program of this caliber. Furthermore, since that review appeared, we have come up with some subtle and powerful features which deserve some detailed attention.

First let us look at how the program handles color. The initial impression is that for a multicolor sketchpad, color has been somewhat restricted. During normal operation, only three out of the sixteen possible colors plus the common background color can be displayed at any time. If you should change one of these work-



At top of page are Art Huff's Night Rider *and* Cats; *above is* Night Cats, *a Wayne Schmidt demonstration of* Paint Magic's *screen-merging and section-masking capabilities. Transpose mode was used to relocate image.*
READER SERVICE NO. 218



Wayne Schmidt's **Country Home,** *drawn with* **Paint Magic,** *provides an illustration of how a very detailed graphic image can be developed. . .*

ing colors, then every occurrence of that color on the entire screen will also change. If this were the final condition, the program would be truly limiting the color capabilities of multicolor mode.

Fortunately, provision has been made to get around this limitation. The key is the Color Mask feature. The important thing to remember when using the color mask is that it only applies to *Paint Magic*'s working colors 2 and 3. The user should keep this in mind when creating a screen image.

The color mask feature allows areas originally colored with working colors number two and three to be selectively painted in any of the sixteen available colors. As a result, *Paint Magic* allows up to two colors to be uniquely defined in each of the 1,000 character cells. The colors normally associated with color memory are restricted to a single color for the entire screen. Although the overall color flexibility was not as great as with some of the other multicolor programs, we found the selective mask capability to be a very powerful medium for artistic expression.

Paint Magic allows a second screen to be merged with a first screen under controlled conditions. Sections of the screen can be masked to locally restrict the merge. The results can be rather effective. The *Night Cats* picture was created when Wayne Schmidt merged two of the sample paintings supplied with the program disk, *Night Rider* and *Cats*. Actually, several features of *Paint Magic* were brought into play to achieve this result. These included the Transpose feature to relocate and shrink the image.

The rays feature, as implemented in *Paint Magic*, is not limited to the production of radial lines. It may be applied in conjunction with the geometrics as well. Thus it is possible to create radial patterns of rectangles, circles, or whatever.

The zoom feature is not restricted to a small section of the image. When finished with a particular area, simply move the zoom cursor to the edge of the field. The entire screen scrolls to another part of the image while still in zoom mode. On top of all this, all of the geometrics will still work in zoom mode. This was the only package which included this last capability.



. . .from a bare bones sketch (a technique suggested for detailed work).

Paint Magic employs a rather novel file structure. The picture files are saved as a BASIC program with an appended machine language loader. To display a *Paint Magic* screen, simply LOAD and RUN. Unfortunately, this may cause difficulty when trying to call up an image from within a BASIC program. The *Paint Magic* loader program loops back upon itself after the image is displayed. There is no way to break out of this loop without affecting the image file or the machine language loader. The manual did not give details of the image file structure.

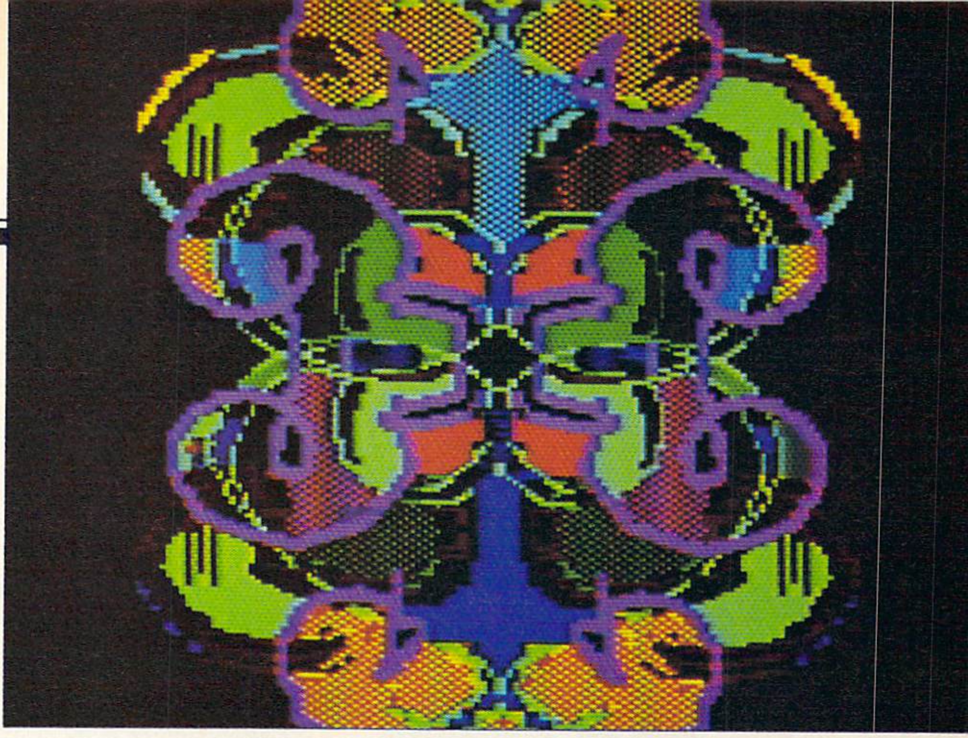
Fortunately, we have been able to analyze the *Paint Magic* image files. Details will be presented with the discussion on the image file structure of all these packages, scheduled for next month. We will include details on how to transfer the various multicolor files for print by the *Picture Perfect* program as well.

Finally, we would like to call your attention to *Country Home* by Wayne Schmidt, which was created with *Paint Magic* for this report. We have also included an intermediate stage of this image to illustrate how a complex image can be built up from a rough sketch. Actually, this technique is recommended for detailed work with all of these drawing packages. The precision with which the user can position the pen on the full-sized image is limited by the resolution of the input device and steadiness of hand. In addi-

tion, individual pixels are difficult to distinguish on the color television or monitor. The latter is particularly true when using colors of related shades for subtle effects.

The recommended approach is to rough out the image in normal mode. Details are then added a pixel at a time in zoom mode. The process is time consuming, but the results will be well worth it. For this reason we placed strong emphasis on the zoom feature in each package. Remember, unlike the traditional oils and canvas, the computer allows earlier stages of an image to be saved for future revision.

We found that the user was likely to expand his abilities with extended use of *Paint Magic*. The program's speed of operation enhanced creative activity as familiarity grew. □



Mirror ***by Wayne Schmidt indicates some of*** **Koala Painter*****'s capabilities.***
READER SERVICE NO. 219

Program Name: *Koala Painter*
Type: Multicolor 160 by 200 pixel
Input Device: Custom Touchpad
Price: $99.00; Includes hardware and software.

Koala Technologies Corp.
3100 Patrick Henry Drive
Santa Clara, CA 95050

The Koala Pad Touch Tablet is a four and one quarter inch square sensitive tablet for converting surface pressure to a signal which is readable by the computer. Simply apply pressure with a finger or stylus and, with the proper software, the computer can sense its position on the pad. From the computer's viewpoint, the tablet actually looks like a pair of mechanically cross-linked paddles or an analog joystick. The output of this type of joystick is directly proportional to the relative angular position of the shaft. A continuously varying signal is generated as the stick is manipulated. By comparison, the Atari type joystick normally used with the Commodore 64 generates eight possible digital output signals. The tablet has two pushbuttons, located at the top, corresponding to the individual paddle pushbuttons.

Included with the Koala Pad is the *Koala Painter* drawing program. This software, when used in conjunction with the touch tablet, allows you to draw with a finger or stylus just as with a pencil and paper. The difference is that the results are displayed on the screen and stored in the computer's memory rather than on paper.

Koala Painter is not just a simple demonstration program which was thrown in with the hardware. It is a very powerful and effective color drawing package. The screen photographs of Wayne Schmidt's *Samurai* and *Mirror* are a very good indication of the program's possibilities.

This is the most user-friendly of all the programs we looked at. At the start, the user is presented with a graphic menu of all of the program's features. The upper two thirds of the screen displays the program's commands. Each command is illustrated by a related sketch and a single word description. You simply move the stylus on the tablet till the arrow points to the desired feature. A press of one of the tablet's buttons causes the selected function to blink. A deft move of the stylus to the bottom of the tablet and another press of a button and you are on the drawing screen ready to carry out the selected command. The sequence of events is not nearly as complicated as it sounds. After about fifteen minutes of this, the operation became so natural that it hardly required any thought.

In addition to the conventional draw mode, you may select from individual or end to end lines, boxes, circles, and rays. All the solid shapes could be done in outline or filled. The drawing of circles was limited to the perfectly round variety. You cannot produce an ovular shape.

Koala Painter allows you to store a completely independent image in the computer's memory. This feature is fully supported in disk mode. Simply swap screens and load in the second image. The availability of a second screen

is a very powerful tool when used with the copy command.

The copy command has two operating modes. It memorizes a selected portion of the screen for reproduction any number of times. It also permits the transfer of a part of an alternate screen. This feature enables you to create a set of screens with standard objects which can be used to fill out an overall image.

Koala Painter's zoom implementation is one of the best we have seen. The zoomed image is displayed on the lower half of the screen while the actual size image is displayed above. All changes made in zoom mode are instantaneously displayed on the main image. In addition, the zoom mode has its own color menu.

Getting back to the main menu, the central portion of the screen offers a selection of eight brush types and sizes. However, the user cannot create his own brush designs. Color selection is done on the bottom portion of the menu. Simply move the pointer to a color and press a button. The border displays the color in use. The color menu also allows the selection of a checkerboard texture pattern instead of the solid paint color. Any combination of the sixteen available colors can be used. The user cannot create his own texture patterns.



Computer graphics are equally capable of representing the sordid side of life, as is proved by Wayne Schmidt's* Samaurai *(drawn with* Koala Painter*).

As we saw in the accompanying article on bit mapped graphics, up to four memory locations are involved in the setting of color. A good measure of the program's power is how well it kept track of color. As a multicolor package, the *Koala Painter* allows for 160 horizontal pixels in an image. Up to four different colors can be displayed in a single character cell, the size of four horizontal by eight vertical pixels. Three of these colors can be uniquely defined for each of the screen's 1,000 character cells. The fourth color is common to all the cells. The program took care of color with very little attention from the user. It seemed to check each cell for which colors were available at a given time. If you tried to put a fourth color into a cell, the program would change the one which would have the least impact on the image.

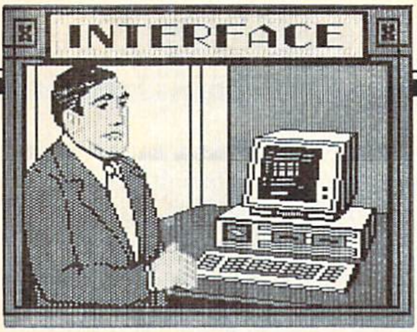
The "xcolor" feature allows the user to instantaneously change all the occurrences of a particular color, on the entire screen, to any other color. A hidden subtlety of this feature, not explicitly stated in the manual, was its use for changing the background color.

Presently, *Koala Painter* lacks a printer dump feature. Fortunately, *Picture Perfect* from KT Software fills this need. We have included a review of this product with this report.

The mirror mode always replicated the image in all four quadrants. We would have liked to see the option for horizontal or vertical operation with this mode.

When disk mode is activated, the directory is scanned and all the names of Koala files are displayed as part of a dedicated disk menu. All disk operations are neatly labeled. Operation is very similar to the main menu. The file structure is described in a supplementary sheet, which includes a loader program, available from the manufacturer.

Overall, we rate the Koala Pad with *Koala Painter* as excellent in ease of use, a fine choice for a beginner as well as young children. *Koala Painter* is just one of the programs available for use with the touchpad. (Koala offers a number of other software packages for use with the Koala Pad.) □



Example of a Picture Perfect *dump.*
READER SERVICE NO. 220

Program Name: *Picture Perfect*
Type: Multicolor Printer Dump
Price: $30.00
K. T. Software
P. O. Box 4943
Huntsville, Alabama 35815

Picture Perfect is not a graphics processing package. It is a fully featured printer utility for converting Koala format screen images to hard copy on a dot matrix printer. The resulting printout displays a Commodore multicolor image in gray scale.

As it turns out, it is a fairly simple matter to convert the disk files generated by any of the graphics packages to Koala format. The conversion is so simple that we were puzzled by the absence of the routines in this package. It seems that the manufacturer needlessly limited this product's market by addressing only Koala format files. In any event, we performed the file conversions ourselves. Some of the sample printouts on these pages were actually imported from *Paint Magic*.

Printing a multicolor image involves more than a bit-by-bit dump. If color is not accounted for, the entire image may end up as a uniform black mass. A proper printout will assign different dot patterns to each of the sixteen colors. If the patterns are properly chosen, the result would be similar to displaying a multicolor image on a black and white television.

Determining the colors of a Commodore 64 multicolor image is not a trivial task. The color information for a particular pixel can come from one of four locations. The program must check each pixel and assign the proper pattern no matter how the color is assigned. *Picture Perfect* does all this and more.

Picture Perfect includes a full set of bit patterns for each of the sixteen colors. It also allows the user to adjust all of the dot patterns to his own needs. This powerful feature makes the program more than just multicolor screen dump. It allows the user to access the much higher resolution available with a dot matrix printer. Remember, a multicolor screen image consists of 32,000 pixels (160 horizontal by 200 vertical). By assigning a pattern of up to sixteen dots (four by four) to each pixel, *Picture Perfect* converts a multicolor image into 512,000 dots.

The ability to select patterns is a very powerful tool. It allows the user to adjust the gray scale for best results with a particular printer and image. Variations in printer and ribbon conditions can thus be taken into account. Contrast between adjacent colored areas may be optimized.

By careful selection of bit patterns, the program can become a design tool in its own right. By carefully selecting the dot patterns and designing the image to suit, the program becomes an effective pattern design tool. The colors in the original image can then be considered as control codes for a particular pattern.

The actual dot pattern available depends on the particular printer. The version of *Picture Perfect* which we tested supported the Epson RX/FX-80, Gemini 10X, Okidata 92, Prowriter and the Commodore 1525/MPS-801. With the Epsons, the user could choose from a two by four or four by four pixel pattern. The Gemini supported only the four by four pattern. The Okidata and the Prowriter worked in two by four mode. The Commodore printers were limited to two by three dot patterns.



Picture Perfect *works by turning a multicolor image into 512,000 dots.*

A two by three dot pattern only allows for seven distinct shades, although the actual number of possible patterns is 64. By carefully selecting the dot patterns, quite acceptable results were achieved even with highly colored images. Of course the four by four pattern is preferred as it allows for a distinct shade for each of the sixteen colors.

When using a non-Commodore printer, an interface on the serial port is required. The manual suggests either the Buscard, Cardco A, Cardco +G or the Connection. We found that the Micrografix MW-350 with a 4K RAM buffer worked as well. The program's manual recommends setting the interface to transparent mode, giving the program direct control of the printer. We found that using either the Connection or the Micrografix interface in emulate mode, with a Gemini 10X, gave very satisfactory results as well as an unexpected bonus. This actually gives the user a choice of an additional output format. In transparent mode, the images are horizontally oriented on the paper. In emulate mode the image is rotated



Wayne Schmidt's **Sergeant Major** *showcases* **DOODLE!***'s color capabilities.*
READER SERVICE NO. 221

ninety degrees to a vertical orientation.

User dot patterns may be saved to disk for later recall. Pattern creation with the built-in editor was straightforward. All of the two by four patterns are simultaneously displayed by the editor. The four-by-four patterns cycled one at a time. We would have preferred to see all of the four by four patterns at once. However, doing so would not have left room on the display for the command menu.

Picture Perfect is very user-friendly. All commands are initiated by one of the function keys. Each of the program modes has its own display screen with all control functions and options displayed. We were very pleased with the operation and performance of this product. □



DOODLE! *is useful for generating custom letterheads and cards.*

Program Name: *DOODLE!*
Type: High Resolution 320 by 200 pixel
Input Device: Joystick, Trackball
Price: $29.95
City Software
735 West Wisconsin Avenue
Milwaukee, WI 53233
414-291-5125

DOODLE! is one of the few high resolution drawing packages available for the Commodore 64. It will allow you to get the maximum bit resolution possible with the computer. The tradeoff is in color flexibility. As we have seen, this is an inherent limitation of the Commodore 64 hardware when working in high resolution mode. As we will see, *DOODLE!* allows for maximum use of the available color in this mode.

DOODLE! is very user-friendly. Each of the ten major operating modes has a dedicated menu. The user can instantaneously toggle between the drawing surface and the current menu by hitting the RETURN key. The main menu, which lists the operating modes, can be accessed at any time by hitting a SHIFTed RETURN. The modes are selected by the eight function keys and Commodore 1 and 2 key sequence.

Once a mode is selected, the corresponding menu screen is immediately available. If the operating mode is changed while a menu is displayed, the menu is changed as well. On returning to the sketchpad, the selected operating mode will be in effect. As a result, the comprehensive instruction manual rapidly becomes superfluous.

The one word that best describes the overall program operation is precision. The operation of the various features is designed to allow precise pixel positioning on the drawing surface. The user can toggle a reference grid onto the drawing surface by simply pressing the G key. It is best to turn off the grid after the cursor is properly positioned since pixel status cannot be changed while under the grid lines.

In place of an OOPS mode, *DOODLE!* lets you memorize the current state of the sketch by hitting the M key. It is a good idea to do this before using any of the automatic operations. The last memorized sketch can be recalled by the R key. This last feature has a built-in safety which requires your confirmation before it is carried out. This will prevent a recall from inadvertently replacing the current screen. It seems to us that a swap feature at this point would have been more effective.

SKETCH MODE

As you might have expected, freehand drawing (or should we say freejoystick) is what this mode is all about. Nine drawing speeds and pencil sizes are available. The pencil may be placed on or lifted off the drawing surface by hitting the fire button. The DEL key will flip the pencil around and let you use the eraser. The left arrow (upper left corner of the keyboard) flips the pencil back. The fire button, DEL and left arrow keys

have similar usage in all of *DOODLE!*'s operating modes. The pencil color can be changed by hitting F2 and then the cursor up down keys. When the desired color is displayed inside the circle, switch back to sketch mode by hitting F1. You can now sketch in a different color.

COLOR AND DOODLE!

In general, color in high resolution mode is not as flexible as in multicolor mode. Only two colors can exist in a given eight by eight pixel character cell. *DOODLE!* makes maximum use of the available color in high resolution mode. Color mode allows you to set both the working foreground and background colors to any of the sixteen possible colors. These are referred to by *DOODLE!* as the paint and paper colors. The important thing to remember when changing colors is that a paint or paper color change will affect the contents of an entire character cell. This will occasionally lead to color conflicts when adjacent pixels of different colors occupy the same character cell.

This does not imply that color in *DOODLE!* is limited. A quick glance at *Middle Earth* by Wayne Schmidt (see page 35) aptly illustrates the point. His *Sergeant Major* and *Pen & Candle* sketches also make effective use of color.

GEOMETRICS

DOODLE! provides for the automatic drawing of lines, rectangles, and circles.

Line endpoints are set by placing two independently maneuverable crosshairs on the screen. The left arrow key draws a line between them. Rays can be generated by moving only one crosshair while leaving the other at a selected position.

In box mode, the four corners and the center are displayed. The height and width can be independently set and the entire box can be moved as well. Associated with the box mode is an Op-Art feature. The manual describes this as "an edge-detect, a negative, and a border flip in that order. The process is repeated over and over until the entire box is filled in" or stopped by a key stroke. The effect is to repeatedly outline a selected shape. You will probably fall in love with the feature the first time you try it.

Circle mode is very similar to box mode with height and width being independently set.



Pen & Candle, ***another Wayne Schmidt*** **DOODLE!** ***The program makes maximum use of color in hi-res mode, not as flexible as in multicolor.***

COPY MODE & STAMP MODE

DOODLE! allows the content of a box to be memorized and moved to another portion of the screen. In the process, the memorized part can be stretched or compressed horizontally and vertically. Rotation of the image is possible as well, but watch out. The pixel aspect ratio on the Commodore 64 is not square. Thus the length to width proportions of the image will change.

Stamp mode is a simplified version of Copy mode. Up to nine stamps can be memorized at one time. Each of these can then be placed elsewhere any number of times. Stamps can be rotated, mirrored, reversed (negative image), and changed in size.

DOODLE! does not have a texture mode; that is, brush patterns cannot be defined. However, a similar effect can be had by using the stamp mode.

LETTER MODE

Text, and for that matter any of the Commodore characters, can be placed anywhere on the screen. In addition, the text can be rotated, stretched, broadened, and printed in reverse. As each letter is typed, the cursor is positioned for the next character. This is true even for rotated letters. Thus vertical messages can be easily typed.

ZOOM MODE

The enlarged zoom area can be scrolled to any part of the picture by moving the zoom cursor to the edge of the screen. The cursor keys are for rapid movement to other parts of the image while in zoom mode.

DISK MODE

DOODLE! allows images to be saved and loaded from disk. A directory of only the *DOODLE!* files is displayed when disk mode is entered. All *DOODLE!* disk files begin with a DD.

PRINT MODE

Included with *DOODLE!* is a

very handy hi-res screen dump. It allows images to be printed one dot per pixel or two dots per pixel. The latter results in an image with four times the area of the former. The actual size is seven by nine inches with a Gemini 10X printer.

Printer mode can be configured to work with a variety of printers including the Commodore 1525, most Epson or Star Micronics, C. Itoh 8510, NEC 8023 and Okidata with graphics. The printer should be connected to the Commodore disk drive port with an interface that can be set to be completely transparent. We have used this feature with a Commodore 1525 and a Gemini 10X. The latter was interfaced with a Tymac Connection and the new Micrografix MW-350 with a 4K RAM buffer. We had no problems running the interfaces in either emulate or transparent mode.

The printer feature does not distinguish colors. Pixels are printed as either on or off. Keep this in mind when creating images for hardcopy use.

DOODLE!'s wide array of features, hi-res graphics, powerful text handling and built-in printer routines have made it very useful for generating custom letterheads, cards, invitations, etc. Virtually all of our local Commodore user groups use *DOODLE!* for creating all of their stationery. *Wedding Invitation* by Wayne Schmidt is just one example. There is even a commercial Commodore-related magazine *(Info 64)* which produces a large part of its layout with *DOODLE!*. □

Graphics programs covered next month:

- *Flexidraw*
- *Computer Crayons*
- *Supersketch*
- *Flying Colors*
- *Sorcerer's Apprentice*

Plus a look at C-64 character graphics, complete with program

Program Name:	*Peripheral Vision*
Type:	Multicolor 160 by 200 pixel
Input Device:	Light Pen
Price:	$39.95 Peripheral Vision $39.95 Light Pen with demo software $59.95 Combined Package

Futurehouse
P. O. Box 3470
Chapel Hill, North Carolina 27514

Peripheral Vision is one of the two packages which used a light pen as an input device. It is the only multicolor package so equipped.

The light pen allows for the direct response we have grown accustomed to by working with pen and paper virtually all our lives. Its use offers an immediacy of input and interaction not possible with any other input device. Unfortunately, the use of a light pen with a video display is not without penalty. Care must be exercised in positioning the system components to minimize arm and hand fatigue. The vertical orientation of the video monitor works against this elusive goal. The visual proximity to the monitor, needed for its use, may be fatiguing to many individuals.

Satisfactory results with the hardware are difficult to achieve at low costs. We found the accompanying light pen to be somewhat lacking in selectivity for some of its assigned tasks. This was verified by the simple expediency of using the Inkwell Systems light pen with the *Peripheral Vision* software. Virtually all of the problems we had encountered were eliminated by this substitution. However, the Inkwell Systems device and software sell for nearly three times the price of this package. As it turned out, the *Peripheral Vision* software exceeded the capabilities of its associated light pen.

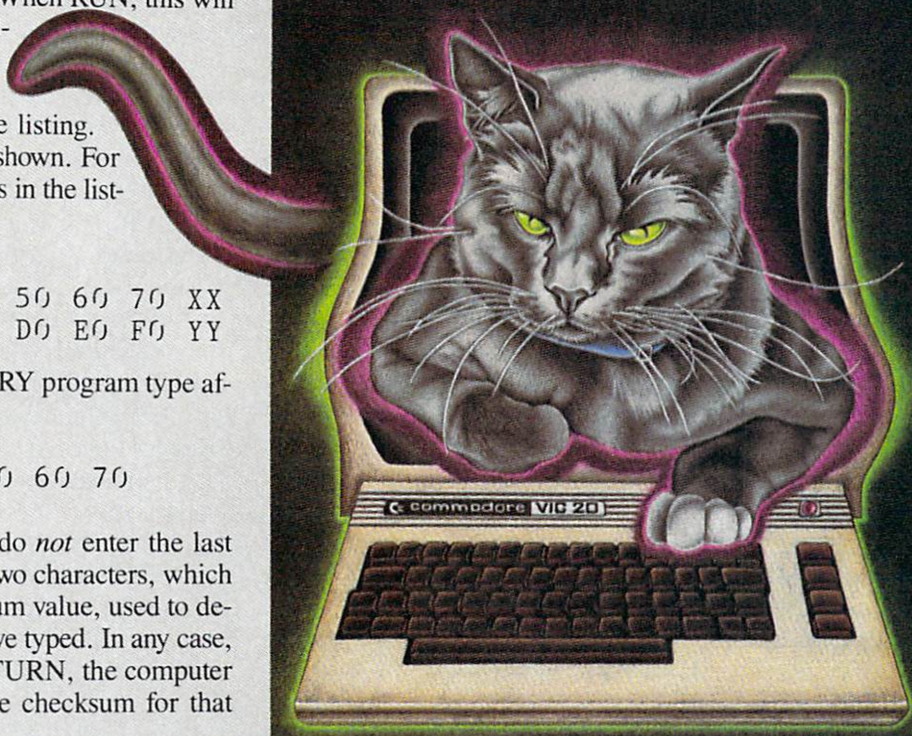
In all fairness we should point out that for the most part we worked with preliminary versions of the *Peripheral Vision* software. A total of four versions before we finished this report, to be exact. In each case, a number of bugs were eliminated with each revision. At least one significant feature was added as well. We have every reason to expect that by the time this report is published some three months hence, *Peripheral Vision* should offer some fairly impressive performance.

The most significant shortcomings of the Futurehouse light pen was the lack of an on-pen switch. This switch would be used to signal the computer when the pen is properly positioned for the desired task. To account for the lack of a pen switch, the software checks for depression of either the Commodore or f7 key. The requirement of keeping the video monitor in close proximity to the computer negates some of the advantage of the extensive onscreen light pen-driven menus, provided by the package. These menus displayed all of the available functions as easily recognizable icons. A total of three menus are available depending on operating modes. These are normally onscreen at all times, but may be toggled with function key f3 to allow full screen sketching.

The software did not include the ability to adjust the light pen offset after the initial setup (see the introductory discussion on light pens). The need for adjustment was felt on a number of occasions. The only way to correct

Continued on page 60

THE VIC* 40-COLUMN OPERATING SYSTEM



© Kat Reviaska 1984

TURN YOUR VIC 20 INTO A PET

BY PETE LOBL

*8K EXPANSION REQUIRED

The *VIC 40-Column Operating System* is a PET-emulating, 40×25 character generator for your VIC 20. A minimum of 8K RAM expansion is needed for its use. The program will work successfully for both tape and disk users alike.

This article is divided into two parts. The first contains the entry program which allows you to enter the all-machine language *VIC 40* in normal BASIC. The second portion of the article concerns the actual *VIC 40* itself, with a BASIC demo explaining some of the program's power.

ENTERING AND SAVEing A WORKING COPY OF VIC 40

The listing of the *VIC 40* is in hex format, almost like a BASIC listing. To enter the *VIC 40*, simply type each line of hex as you would a BASIC program. First, type in and RUN the ENTRY program. When RUN, this will display a period followed by a colon, ended with the blinking cursor. At this point type in the hex digits exactly as they appear in the listing. Make sure to include spaces where shown. For example, two lines may look like this in the listing:

```
.:2888 00 10 20 30 40 50 60 70 XX
.:2890 80 90 A0 B0 C0 D0 E0 F0 YY
```

To enter these lines into the ENTRY program type after the .: prompt

```
2888 00 10 20 30 40 50 60 70
```

then hit RETURN. Note that you do *not* enter the last two characters in each line. Those two characters, which are whited out, constitute a checksum value, used to debug any mistakes which you may have typed. In any case, after you enter the line and hit RETURN, the computer will respond with a line stating the checksum for that line. It will appear like this:

```
CHECKSUM = ZZ
```

The zz will be a two-digit number which will correspond to the numbers you entered. Compare the number the computer gives you to the number printed in the magazine. If they are not equal, then reenter the line you just typed in and make sure everything is typed correctly. If the number given by the computer and the number printed in the magazine are equal, then all is correct and you may proceed to the next line. Follow this procedure for all lines, comparing the computer's checksums to

those in the magazine. Be prepared to do a *lot* of typing, but try to concentrate as one mistake could ruin the entire effort!

When you are finished entering the *VIC 40,* type the letter 'f' at the .: prompt. You will then be asked whether you want *VIC 40* saved to disk or tape, depending on which unit you own. If all goes well, when the dust has settled a small, quick-loading, complete *VIC 40* will be on your disk or tape!

NOTE!!!!!!

If for some reason you cause the ENTRY program to crash, or generate an error message on the screen, type

```
GOTO 50
```

followed by a RETURN. This will put you back into the entry program with memory left intact. Don't be afraid that anything bad has occurred, just reenter the line that caused the error and continue along normally.

After the *VIC 40* is SAVEd onto disk or tape, you will need information on how to properly use it to meet most of your needs. The next section of this article is devoted entirely to that purpose and assumes you have a working copy of *VIC 40* saved already. All ready? Here we go!!!!

OPERATION AND USE OF VIC 40

The *VIC 40* is an all-machine language program which cannot be loaded and RUN as BASIC programs are. A special loader is needed, and one is provided for you in the listing section (marked LOADER). It will work with either tape or disk, as the *VIC 40* does.

This section is only for tape users; disk owners, skip to the next paragraph. Tape users, take note! Make sure that the loader comes before the main *VIC 40* program. Because the tape has to search sequentially for programs, unlike the disk which can go anywhere on a surface, programs must follow in the order in which they are going to be run. The loader has to go right BEFORE the main *VIC 40* program in order for a proper LOAD to occur. Any other sequence will not work!

After LOADing and RUNning the loader, the main *VIC 40* program will be found, loaded, and executed. If all goes well, you should see a display screen unlike that which you have seen to date on your ole' VIC 20. A power-up message similar to the one you receive when the power is turned on will be displayed, only now in 40-character across format. In fact, your VIC will now act like a PET in many respects. For one thing, PET screen POKES will now work (32768-33767) along with PET sound generation (59464, 59466, 59467). Even the trusty upper/lower case switch (59468) will work, allowing any old PET BASIC software to work with relative ease on your expanded VIC!

Some of you may now say, "I don't know what those PET POKEs are for; therefore I have no use for them!" If you are in that group, read on to discover functions which the *VIC 40* can perform unrelated to the PET (but which are still powerful). Old CBM/PET users will, however, respect the versatility this gives them!

GETTING SPECIFIC

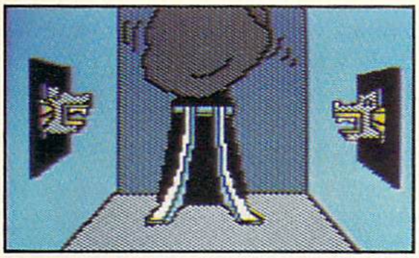
As stated before, *VIC 40* operates almost identically to the way the VIC does on power-up, except that now everything is in 40 columns. One noticeable difference is the presence of a real control key. The CTRL key used to change colors; now it will cause an indefinite pause until either SHIFT key gets pressed. To use the CTRL key to change colors or to turn on/off reverse field, SHIFT-CTRL must now be used. This feature is designed as an aid when LISTing programs, or making the computer wait until you are ready.

Another key with a different function than normal is the RESTORE key. If you need to break out of a dead program, or for some reason you wish to return temporarily to 22-column mode, press the RESTORE key by itself. No STOP key press is necessary to generate the warm-start. To reenter the *VIC 40* from 22-column mode, type SYS 8841 if the power-up message is desired. If no message is needed, and you want the screen colors to remain the same, type SYS 11768. Either way, you will be back in the *VIC 40.* As a rule of thumb, use SYS 8841 if you enter from direct mode, i.e., right from the keyboard. Use SYS 11768 if you enter *VIC 40* from a program, as the *VIC 40 Demo* does.

VIC 40 contains both of CBM's character sets and has the special capability to display both on the same screen at once! To change character sets you must either do a POKE or a PRINT. To go to lower case use POKE 59468,14 or PRINT CHR$ (14). To switch to upper case/graphic mode type POKE 59468,12 or CHR$ (142). The normal way of using the SHIFT and COMMODORE key to change cases does NOT work in *VIC 40,* so please learn the other methods.

Characters are stored in a 4×7 matrix inside *VIC 40*'s memory. Two characters are stored in each byte, up to a total of 128 characters (the normal number of VIC 20 characters). Reverse-field characters are not stored, but are generated on the fly by simply reversing the bits of a stored character, then displaying it on the hi-res screen. Hence, 128, not 256, programmable characters are at your disposal. Remember, if you do design a programmable character, you will have to change an already existing one, perhaps for a game or special business application. The reversed field version of your character will also

Continued on page 64





Adventure-game your way through graphically depicted Aztec ruins.
READER SERVICE NO. 201

MASK OF THE SUN
Broderbund Software
Commodore 64
Disk

While examining an ancient pre-Columbian amulet from central Mexico, you (a noted archaeologist/adventurer) discover a tiny compartment which, when opened, releases a pale green gas. Unfortunately, the gas causes you to lose consciousness for two days.

The doctors, though mystified, are able to produce a pill that will slow you body's rapid degeneration. Your renewed research indicates, however, that the legendary *Mask of the Sun* may hold the cure. Pills in hand, you head for the Aztec ruins of South-Central Mexico in search of the mask.

Like all good adventure games, *Mask of the Sun* creates a world within your computer that seems real enough to demand your exploration. With the aid of Raoul, your Mexican guide, you must search ancient Aztec temples and pyramids for the golden mask. Along the way you encounter poisonous snakes, mysterious stone idols that spring to life, blood-stained altars, and enough false trails to keep you busy for quite a while.

The story is enhanced by colorful comic book graphics and sparse but effective use of sound. Being able to hear gunshots, a snake's hissing, the sweet music of a flute, or even the grinding of moving stone slabs adds dimension and realism to the adventure.

Commands are entered through the keyboard in either complete or truncated sentences, and multiple commands such as "ENTER JEEP, DRIVE WEST, THEN LEAVE JEEP" can be entered simultaneously as long as they all fit on the 37-character command line. The program has an adequate vocabulary so you rarely have to struggle with phrasing in order to make yourself understood.

In addition to a good working vocabulary, the program also has a great sense of humor. Kick Raoul, for instance, and Raoul kicks you back. Kick the jeep and the computer responds with "Ouch! You really should control yourself." At another point the computer puts you in a potentially life-threatening situation and, just as quickly, saves you with "No, no, just kidding."

Unfortunately, *Mask of the Sun* is not without a couple of faults. First, some of the animated sequences take too long. For example, when you give the command to DRIVE NORTH you're treated to ten or fifteen seconds of changing road scenes after the disk has been accessed. Although this does add realism to the game at first, eventually it becomes slightly annoying. A similar problem occurs in the maze of tunnels where every turn you take takes several seconds to complete.

The program's only other fault is that even novice adventurers should be able to complete the game within a couple of weeks. Like a good book, once *Mask of the Sun* has been completed and you know the ending, it's liable to sit on the shelf for quite a while before you give it another try.

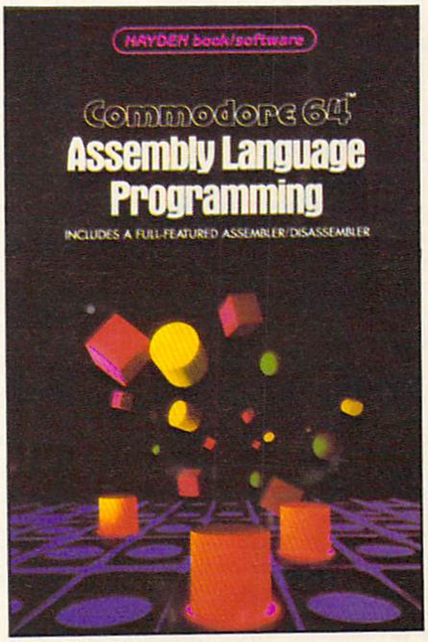
Broderbund Software, 17 Paul Drive, San Rafael, CA 94903 (phone: 415-479-1170).

—Bob Guerra

C-64 ASSEMBLER
Hayden Book Company
C-64
Cassette

Hayden's *C-64 Assembler* ($29.95) is a tutorial which shows you the basics of assembly language programming. Complete with a 225-page book, the program tape includes an assembler and a program to tutor you on converting numbers from one base to another.

Assembly language is the programming language closest to the computer's own tongue. Although assembly instructions have mnemonic labels, like BASIC commands, each instruction has a



Some BASIC knowledge required.
READER SERVICE NO. 202

limited effect on the computer, like putting a number in a memory location, jumping to a specified memory address, or comparing two numbers. Your assembly language program will be more detailed than a similar BASIC program but much faster.

C-64 Assembler is labeled "A complete course for the absolute beginner." Let's qualify that a bit. By page 4 of the book, you will be introduced to four assembly language instructions and shown a simple program. By page 7, you will add three new instructions and a second program. You do not need to know anything about assembly language, but you'd better be familiar with BASIC programming concepts and with your computer's hardware. Someone capable of writing non-trivial BASIC programs is probably at the right level of absolute beginnerness.

Although this is a full-featured assembler, it is designed as a learning tool, not as a productivity tool for the accomplished assembly language programmer. The assembler itself is written in BASIC, which limits its speed somewhat. And the book uses a tutorial approach; no reference manual or glossary is provided.

Further, the only way to save or load an assembly language program is by using the machine language monitor included with *C-64 Assembler*. It allows you to save your program after specifying starting and ending addresses (memory locations). It also lets you display the contents of a block of memory on the screen, directly modify memory a byte at a time, and move chunks of memory around. As you become accomplished at assembly language programming, a monitor is an essential tool for debugging.

This monitor will also convert an assembly language program into BASIC data statements, which are then appended to the end of the *C-64 Assembler*. Writing machine language subroutines for your BASIC programs becomes a snap, although, since *C-64 Assembler* uses the cassette buffer, saving data statements on tape is a tricky business.

The 6510 CPU has an instruction set made up of 56 mnemonics, described in detail in Appendix 2 of the *C-64 Assembler*. In the standard instruction set, each mnemonic is three letters— like JMP to jump to a specified location in memory. That mnemonic is usually followed by an operand, the memory address or number to which the mnemonic refers.

C-64 Assembler uses modified mnemonics for its instructions. It combines the three standard characters with one or two additional characters that indicate the addressing mode. Different assembly instructions have up to eight addressing modes to tell the computer how to find locations in memory.

These modified mnemonics may help a beginner to learn assembly language; they certainly force you to be aware of the addressing mode for each instruction. But they are not standard. You will learn JMPIA as an instruction. If you move to another assembler, you will only have JMP to work with, and you will have to figure out how to modify the operand to specify addressing mode.

Introducing the most important instructions first, the book covers new topics only as they are necessary to describe more instructions. You will not find a discussion of number systems until chapter 4 when mathematical and logical operators are discussed. This is a nice approach for those of us who like to jump right in. You'll find yourself doing first and understanding later. *C-64 Assembler* takes a radically different approach from other tutorials, most of which begin with chapter after chapter discussing number systems and hardware architecture.

User involvement is virtually demanded by this approach. The book expects you to type in the sample programs and gives you exercises (29 in all) to test your understanding. Flowcharts are put to good use to show how program instructions move in logical sequence. Charts that show the actual contents of each register in the 6510 as it runs a short program are also used as learning aids. At any time, you may ask for a listing of your program. Each assembly instruction will be displayed with its machine code counterpart

and its address in memory.

By chapter 5, you'll have covered most of the basics and will be ready for labels. A label is a word that can be included in your assembly language program to mark a subroutine. When you need that subroutine, just jump to the label and the assembler will find the subroutine.

C-64 Assembler also allows you to use macros within your program. A macro is a series of instructions with its own label. Just like a subroutine—so far. The difference is that when the assembler turns your program into actual machine code, it inserts the full macro in each location where you have referred to it. Your assembled program will be longer, but it will run faster than if you had used a subroutine that the program must jump to and return from.

The number of labels and macros you can use is limited by the assembler so major assembly language projects may be hard to complete. But the book does describe how to modify the BASIC listing of the assembler to allow more labels and macros.

Chapter 7 shows you how to use machine language subroutines that are built into both BASIC and the Kernel. Not only are those routines already written for you, but they are also coded to run as fast as possible.

Appendix 3 gives a detailed list of the C-64's memory locations and the functions each one controls. About 700 different addresses are listed (Commodore's *Programmers Reference Guide* lists only half as many) and cross-referenced to VIC and PET computers. Noticeably missing are the addresses for the VIC, SID, and CIA chips.

Number systems—binary-coded decimal and hexidecimal—are described in Appendix 1. The second program on the tape, 64 Binary/BCD/Hex Tutor, complements this appendix. You can choose to display a number in the various formats, or you can do exercises that require you to convert between decimal and hexidecimal, binary, or binary-coded decimal.

If you have the rudiments of BASIC programming down pat and would like a good tutorial introduction to assembly language programming, Hayden's *C-64 Assembler* may be just the trick. When you have finished the tutorial, you will still have a tool that is adequate for writing and debugging assembly language subroutines and short programs.

Hayden Book Company, 50 Essex Street, Rochelle Park, NJ 07662 (phone: 201-843-0550).

—Richard Herring

THE CLONE MACHINE
Micro-Ware Dist. Inc.
C-64
Disk

If you have read every article you can find on how your 1541 drive works and it still does not make any sense, *The Clone Machine* ($49.95) can help. Or if you worry about not being able to back up protected software as you watch your five-year-old head toward the computer with a bowl of Lucky Charms sloshing in his hands, *The Clone Machine* may be your answer. This software offers you several ways to examine directly the contents of a disk, make modifications, and attempt to make backup copies.

When you just want to copy a few files, *The Clone Machine* lets you hit a single key to select the ones to copy. With one drive, you will have to switch disks for each file. If you opt to copy a whole disk, you won't be bothered with



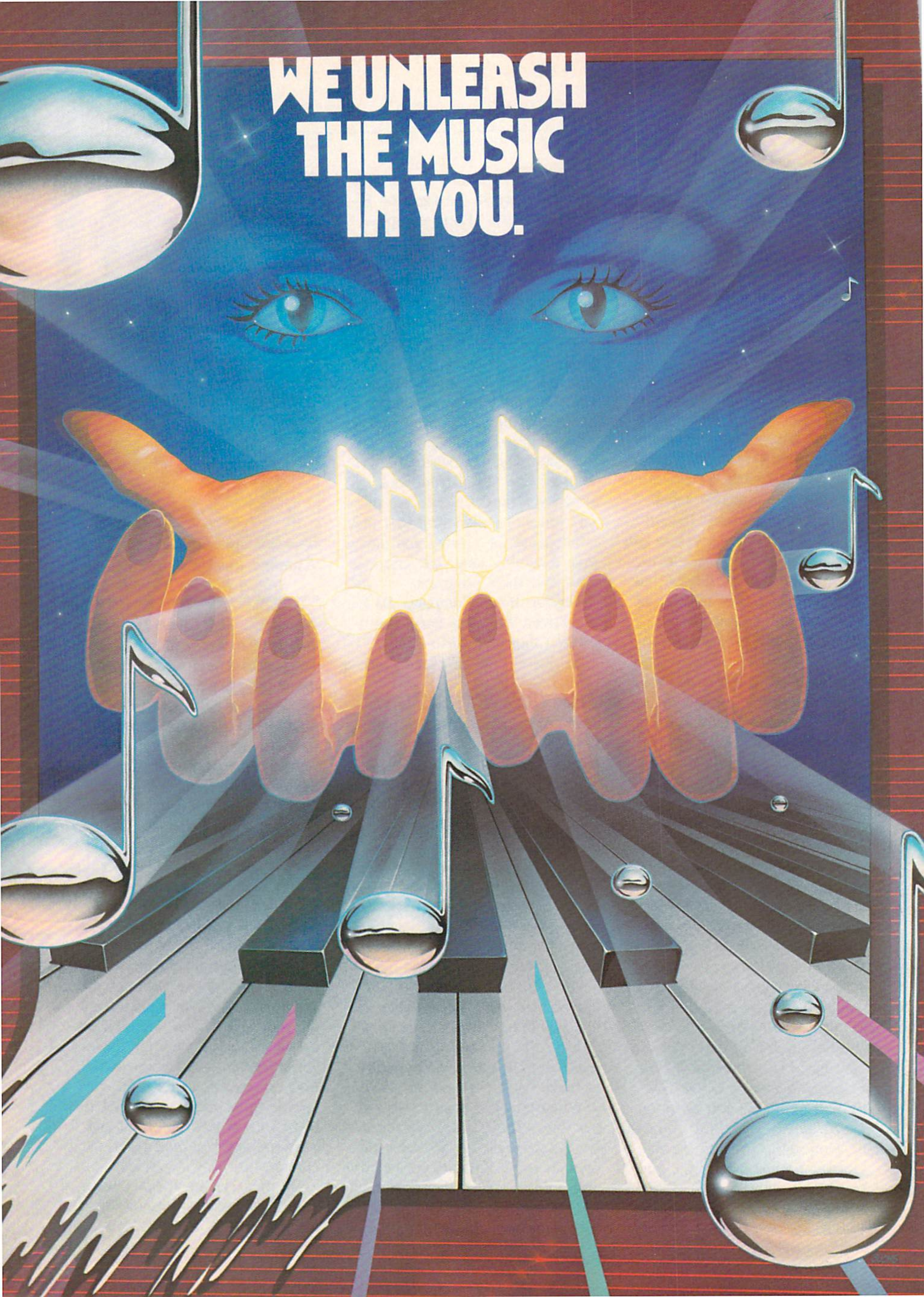
Examine, modify, or, yes, copy disks.
READER SERVICE NO. 203

filenames. *The Clone Machine* will read in every block from the original disk and write it out to the copy. This requires six disk swaps unless you tell the program to copy only a range of tracks.

As you begin any copy operation, the program will give you the option to format (initialize) a new disk. Before it does anything destructive (like format a disk or overwrite a file) it will ask you to double check that the correct disk is in the drive. During any copy operation, the screen will tell you exactly what is happening and how far along the process is.

Another option, the most educational part of this program, lets you read individual blocks from the disk and display the contents on the screen. Data from a block are shown in a 16 by 16 matrix. You can choose to have the data appear in hexidecimal format (00 to FF) or in ASCII format and overtype any byte you'd like to change.

By pressing the function keys, you can have the program show the next block, the last block, re-


WE UNLEASH
THE MUSIC
IN YOU.



Sight & Sound Music Software can do more than turn your Commodore 64 into a music synthesizer.

Deep within every Commodore 64 lurks the unexpected. A versatile music synthesizer. A music video machine. Even a three-track recorder. All waiting for you to control. All you need is Sight & Sound Music Software to make and record sounds that rival those of a real synthesizer. Including brass, strings, piano, guitar, electric bass, drums, funky clav, space wah and other sounds.

It's so easy that anyone can create and record music.

If you can press a few computer keys, you have all the skill it takes to create and play music. They're that music friendly. Our easy-to-follow, fully documented instructions guide you through every program.

And you can let your imagination soar. Creating and recording your own sounds. Your own songs. Your own music. Using the polyphonic keyboard. Playing across a full 8-octave range.

With one-finger, you can control wave form, vibrato, sweep and other special effects.

No one else offers our exciting animated graphics. Or the music of the famous recording artists we offer.

Every Sight & Sound Music Software program features eye popping graphics that explode across the screen.

And you can generate your own colorful graphics. Then, synchronize them to your own music…or to melodies made famous by Michael Jackson, Lionel Richie, Billy Joel and others.

A dozen different ways to have fun with computer music.

Sight & Sound Music Software offers music for everyone's musical interest. Like the Kawasaki Synthesizer that instantly puts a synthesizer's sound and flexibility at your fingertips.

Or, the Kawasaki Rhythm Rocker that creates space sounds and graphics to the pre-recorded rhythms of electronic sound.

Or, even our Music Processor that allows you to create, edit, record, play and even compose your own musical arrangements.

From learning programs to the songs of your favorite artists…we've got it all.

A keyboard that makes easy-to-play even easier.

An easy way to get started making music is to begin with our Incredible Musical Keyboard. It fits over your computer keyboard and allows easy note-by-note play. Included with this optional keyboard is an introductory music software program and two music books. One for the beginner, the other for the more advanced musician.



So, see your local computer software dealer for all the exciting Sight & Sound Music Software. And unleash the music in you.

SIGHT & SOUND
Music Software, Inc.

We unleash the music in you.

Reader Service No. 205

turn to the disk directory, print what is on the screen, or sequentially show the blocks in a file. Being able to look at the contents of a file provides valuable information if you are trying to transfer data between two programs or to recover part of a crashed disk.

Another program on *The Clone Machine* disk shows you the contents of the disk directory. In separate columns, you see the file name, type, starting track and block, file size, and record length (for relative files). You can print a copy of this information, change file names by overtyping them, delete files, and even create them.

The notorious Bit Allocation Map (BAM) is also available for your inspection. A special display uses thirty-five columns to represent tracks and twenty rows to represent blocks in those tracks. Blocks in use are marked in red; unallocated blocks are green, so you cannot print this screen.

The Clone Machine's thirty-eight page manual describes all its features in detail. Six full pages are devoted to the various types of errors which may occur and how to correct them. With all that information, descriptions of things like tracks and blocks, the disk formatting process, and how the drive can find a particular piece of information on the disk are all inadequate.

The instructions suggest using this program and a monitor disassembler to patch programs which are copy-protected. A beginner will not find enough information to allow him to comprehend the process. Since the program prompts you through each step so carefully, it appears that Micro-Ware is more concerned with the copy process than with the education this program can offer.

The latest release of *The Clone Machine* has another program, Unguard, which finds and creates blocks with errors. The instructions say that the program may not load every time—it doesn't. Unguard can search a whole disk or single tracks and give you a report (on screen, not printed) of the type of error in each block. You can attempt to create blocks with errors; the program will try several times, but may not be successful. The supplemental manual, this one ten pages long, does not explain what the errors are; it just tells you how to try to duplicate them.

Micro-Ware Dist. Inc., P.O. Box 113, Pompton Plains, NJ 07444 (phone: 201-838-9027).

—Richard Herring



Create, edit, and animate sprites.
READER SERVICE NO. 204

SPRYTEBYTER

Microtechnic Solutions, Inc.
C-64
Disk

Sprite editors are a common graphic utility for the Commodore 64. They are almost indispensible for anyone working with sprite graphics. Among a crowd of competitors, *Sprytebyter* ($34.95) has several features which make it stand out. It is a professional level package, and may be difficult for beginners.

Sprytebyter can create and edit both monocolor and multicolor sprites. Its "movie" feature lets you see how a series of sprites will look in animation. A bonus program, *The Game Maker*, splices sprite data from *Sprytebyter* into your game program with minimal trouble.

The weakest part of the package is the user's manual. In all fairness, I received only a preliminary manual. However, I had trouble finding information in it. The instructions are unhandily arranged, and are not indexed or cross-referenced. If you dig hard enough, most of the information you need is in the documentation. I would have liked more explanation on some functions, especially on using color with the multicolor sprites.

The basic menu of the editor is the monocolor draw menu. You can draw horizontal, vertical, and two kinds of diagonal lines through the cursor position. Individual points on the 24 x 21 drawing grid can be turned on or off with the space bar. You control the cursor with either the cursor-control keys or a joystick. You can also use the joystick fire button to draw.

As you build the sprite, your design is echoed four ways in the upper right corner. You see the sprite normally, and as it would look expanded horizontally, vertically, or in both directions. One menu option puts the whole sprite in reverse video.

Separate menus are provided for shifting and rotating. You can specify how many spaces to shift the

Continued on page 87



BAM READ & PRINT

FOR THE C-64 AND VIC 20

By Morton A. Kevelson

As dedicated Commodore users, we are well aware that with regard to the disk drive, the term BAM is not indicative of violent destruction. A careful reading of the *VIC-1541 Single Drive Floppy Disk User's Manual* informs us that the Block Availability Map (BAM) is how the 1541 keeps track of which blocks on the disk are in use. It also tells us of the importance of the BAM and the need to keep it updated. It even describes the DOS commands (BLOCK-ALLOCATE, BLOCK-FREE) which allow us to manipulate the BAM.

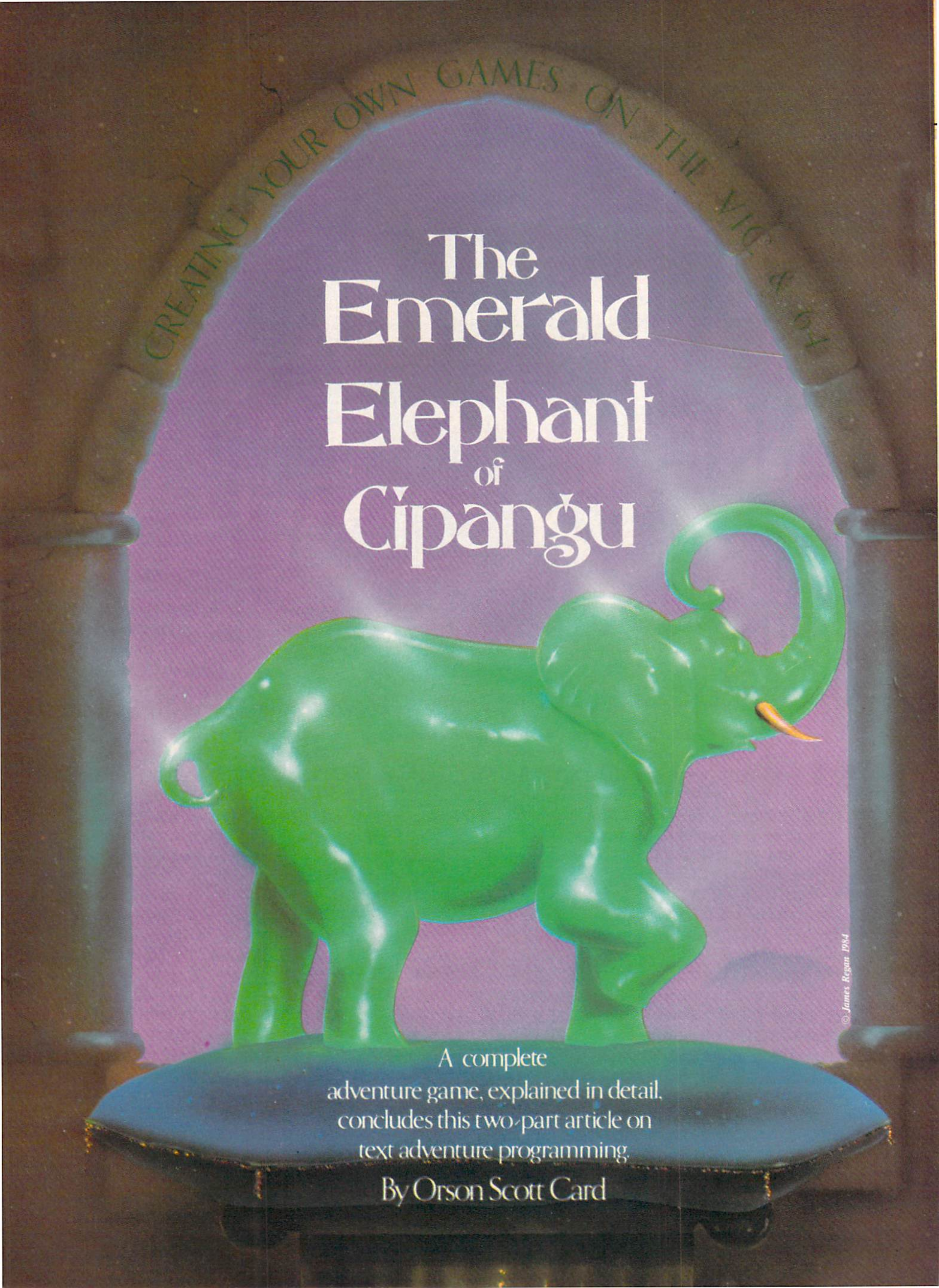
When it comes to a specific explanation of how the BAM is actually written, the manual is somewhat skimpy. The only detail in this regard is found in the table on the bottom of page 55. This tells us that the BAM is a "bit map of available blocks" stored in bytes 4 to 143 of track 18, sector 0 on the disk. This is actually the 5th to the 144th byte of this block. (Remember that we like to start counting at zero with computers.) Some brief footnotes with the table reveal that if a bit is set to one, the block is available and if the bit is set to zero, the block is not available. We are also told that "each bit represents one block," a most profound revelation.

Fortunately, we are not left with loose ends. A program is provided on the Test Demo disk, which is included with the disk drive, for displaying the BAM. If we just take the time we can decipher the code and find out what it is all about. Somehow, with the exception of the most rabidly curious among us, we just never seem to take the time. Besides, the program does seem to work—as long as we are content to live with only half of the BAM displayed at one time. This allows the display to fit on the VIC 20 screen. Besides, we are not really concerned that the display remains visible for only a few seconds. Right?

Well, here is the inside story of what the BAM is all about. We have even provided our own BAM display program which puts the entire BAM on the screen of the Commodore 64. The output can also be directed to a printer for extended study.

The BAM occupies 140 bytes of track 18, sector 0, on a Commodore-formatted floppy disk. The map is actually broken up into 35 groups of four bytes. Each group represents a track on the disk. The first group of four bytes corresponds to track one. The first byte of each group is a count of the remaining

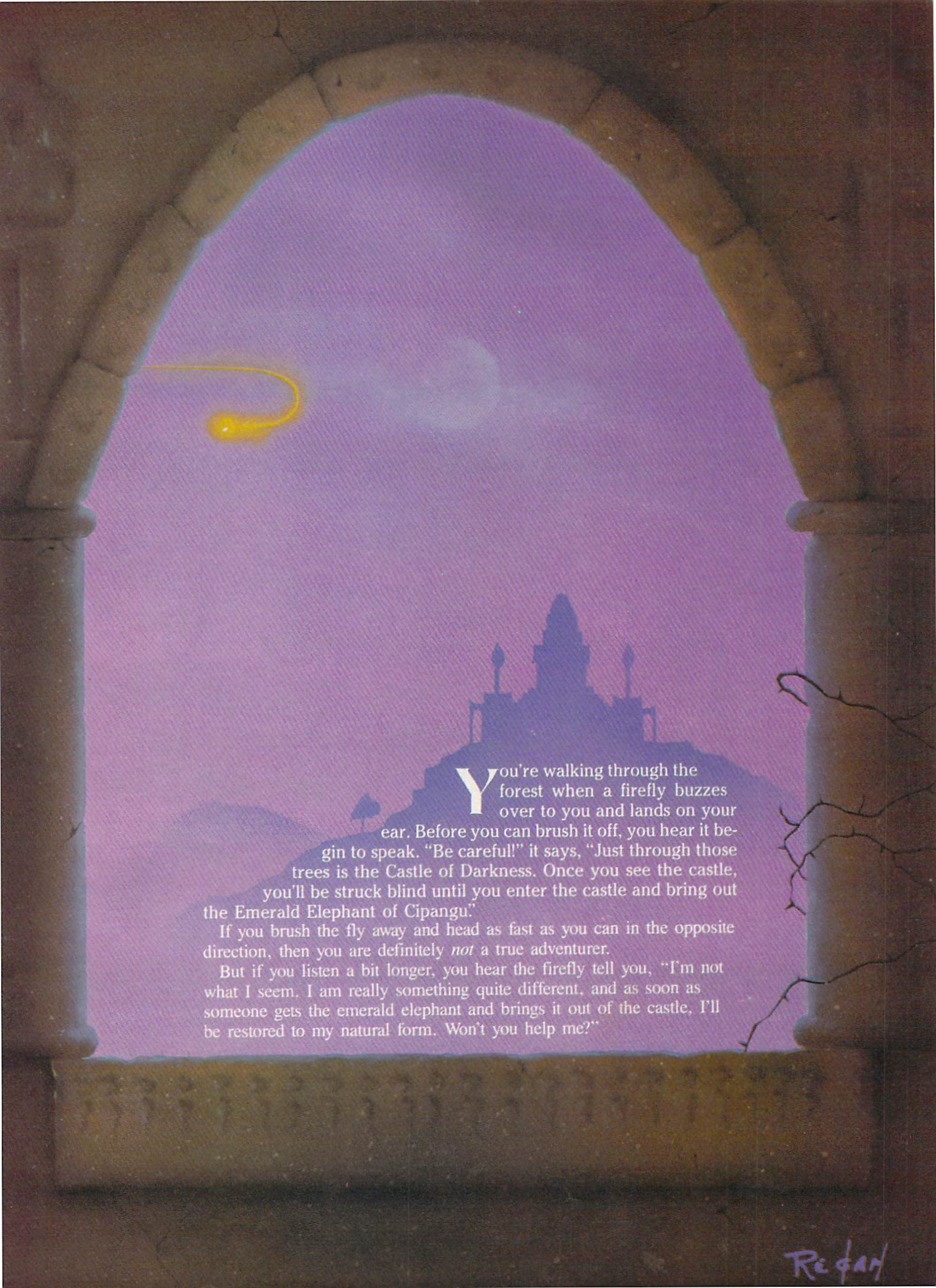
Continued on page 95



The Emerald Elephant of Cipangu

A complete adventure game, explained in detail, concludes this two-part article on text adventure programming.

By Orson Scott Card



You're walking through the forest when a firefly buzzes over to you and lands on your ear. Before you can brush it off, you hear it begin to speak. "Be careful!" it says, "Just through those trees is the Castle of Darkness. Once you see the castle, you'll be struck blind until you enter the castle and bring out the Emerald Elephant of Cipangu."

If you brush the fly away and head as fast as you can in the opposite direction, then you are definitely *not* a true adventurer.

But if you listen a bit longer, you hear the firefly tell you, "I'm not what I seem. I am really something quite different, and as soon as someone gets the emerald elephant and brings it out of the castle, I'll be restored to my natural form. Won't you help me?"

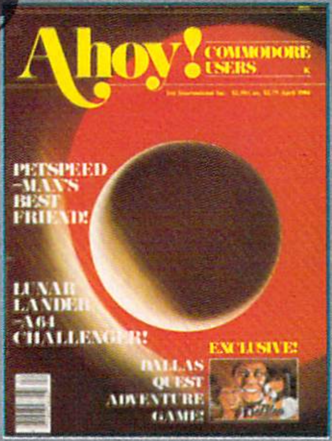
Back Issues of Ahoy!





Sorry—issues #1 and #2 sold out! Limited edition reprints (programs and feature articles only!) available for $4.00 each.



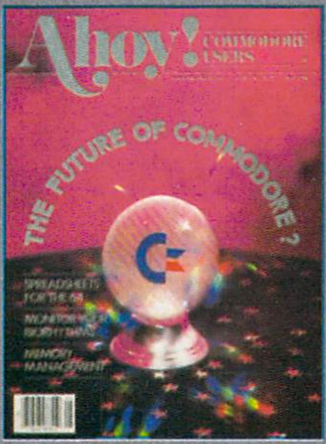


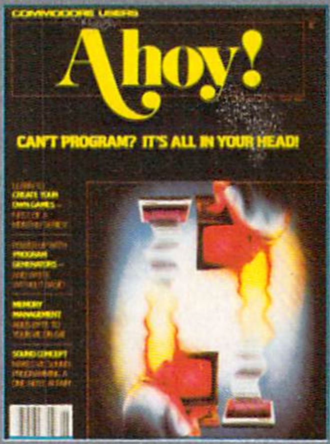
ISSUE #1—JAN. '84 $4.00
The 64 v. the Peanut! The computer as communications device! Protecto's Bill Badger interviewed! And ready to enter: the Multi Draw 64 graphics system! The Interrupt Music Maker/Editor! A Peek at Memory! Programming Sequential Files!

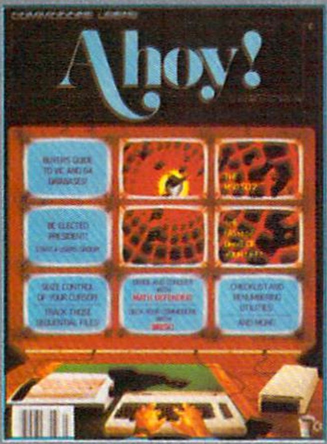
ISSUE #2—FEB. '84 $4.00
Illustrated tour of the 1541 disk drive! Synapse's Ihor Wolosenko interviewed! Users groups! Artificial intelligence! And ready to enter: Music Maker Part II! Night Attack! Programming Relative Files! Screen Manipulation on the Commodore 64!

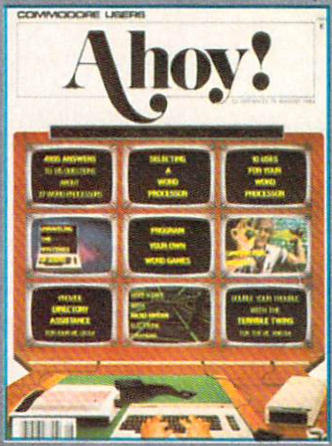
ISSUE #3—Mar. '84 $4.00
Anatomy of the 64! Printer Interfacing for the 64 and VIC! Educational software: first of a series! Commodares! And ready to enter: Space Lanes! Random Files on the 64! Easy Access Address Book! Dynamic Power for your 64!

ISSUE #4—APR. '84 $4.00
Petspeed and Easy Script tutorials! Printer interfacing and educational software guide continued! Lower case descenders on your 1525! Laserdisc! The Dallas Quest Adventure Game! And ready to enter: Apple Pie! Lunar Lander! Name that Star!









ISSUE #5—MAY '84 $4.00
The Future of Commodore! Inside BASIC program storage! C-64 Spreadsheets! Memory Management of the VIC and 64! Educational Software Guide continues! And ready to enter: Math Master! Air Assault! Biorhythms! VIC 20 Calculator!

ISSUE #6—JUNE '84 $4.00
Game programming column begins! Program generators! Rupert on inputting! Memory Management and Educational Software Guide continue! And ready to enter: Post Time for the 64 and 20! Alpiner! Sound Concept for the VIC 20!

ISSUE #7—JULY '84 $4.00
The MSD Dual Disk Drive! Database buyer's guide! The File Sleuth! Creating your own games! Training your cursor! Users groups! Commodares! And ready to enter: Checklist! Renumbering! What's My Job? Brisk! Math Defender! More!

ISSUE #8—AUG. '84 $4.00
Choosing a word processor! Unraveling the mysteries of sound on the 64! Computational wizardry! Creating your own word games! And ready to enter: Micro-Minder! Directory Assistance! The Terrible Twins! Words Worth!

ISSUE #9—SEPT. '84 $4.00
Program your own text adventures! Build a C-64 cassette interface! Video RAM! 10 top word processors reviewed! Word processor applications! And ready to enter: Salvage Diver! DOS! Sound Explorer! The Castle of Darkness!

Send coupon or facsimile to:

Ahoy! Back Issues, Ion International Inc., 45 West 34th Street—Suite 407, New York, NY 10001

Ahoy!

Please Send Me The Following:
_____ Copies of issue number _____
_____ Copies of issue number _____
_____ Copies of issue number _____

Enclosed Please Find My Check or Money Order for $________

(Outside the USA please add $1.00 for every copy)

NAME________________
ADDRESS________________
CITY________________
STATE________________
ZIP CODE________________

And, being stout of heart and, perhaps, a little greedy for reward, you agree to help. You step through the trees and see the castle and—

From then on, you are guided by the voice of the firefly, telling you where you are and what it sees around you. You go to the main gate of the castle, knock on it—but no one answers. There isn't a sound from the castle. How will you get in? Once inside, how will you find the hidden treasure room where the Emerald Elephant of Cipangu is waiting? And if you get into the treasure room, how in the world will you ever get back out?

The game is listed starting on page . Like all text adventure games, it's long—but it's fairly easy typing, since most of the program consists of words that are to be PRINTed on the screen. If you only want to play the game, you can go ahead and type it in—all the instructions are included in the game itself. If you're ever in doubt, just type H or ? for a list of commands.

But if you also want to understand exactly what's going on in the programming, read on. . .

REVIEW OF LAST MONTH'S TABLES

In last month's column, we created the shell of this game by setting up several tables. A table, you'll remember, is an ordered list. For instance, the Room Name Table lists 14 rooms in this order: CASTLE MAIN GATE, MEADOW WEST OF CASTLE, LEDGE EAST OF CASTLE, GROVE SOUTH OF CASTLE, KITCHEN, COURTYARD, GATEHOUSE, ON THE WALLS, GREAT HALL, COUNT'S CHAMBER, TOWER LOOKOUT, STABLES, DUNGEON, and TREASURE ROOM. The order is as important as the names. Because it is first, CASTLE MAIN GATE is room 1; TREASURE ROOM is room 14. By putting these names in the array RN$*(n)*, we can PRINT the name of any room—room 8, for instance, with this simple command: PRINT RN$(8).

The Command Table allows eight different directions of movement: NORTH, SOUTH, EAST, WEST, UP, DOWN, IN and OUT, in that order. There are also other commands: BACK, TAKE, LEAVE, PEER, GOT?, QUIT, HELP, and ?. Each command is invoked by pressing the key for the first letter of the command: N,S,E,W,U,D,I,O,B,T,L,P,G,Q, H,? Any other keystroke will cause the word WHAT? to be displayed.

The most vital table is the Room Direction Table, in the array RD*(n,n)*. There are fourteen rooms and eight possible directions of movement. The present room number, the one the player is in at the moment, is contained in the variable PR; the direction of movement that the player has commanded is contained in the variable CM. If the player is in room 5, the Kitchen, and commands direction 6, which is *down*, he will end up in room number RD(5,6), which is 13, the Dungeon. If the player is in the Kitchen and commands *east*, direction 3, he will end up in room number 9, the Great Hall, for 9 is the value of RD(5,3). (The complete Room Direction Table is contained in Table 1. Rooms 50 and above are not really rooms at all, they are merely illegal-movement messages that are displayed when the player tries to go in a direction that isn't allowed.)

TABLE 1
ROOM DIRECTION

The number after each direction letter tells what room you will end up in (or which illegal-movement message will be displayed) if you command the program to go in that direction.

1—Castle Main Gate
N=52
S=53
E=3
W=2
U=51
D=50
I=53
O=50

2—Meadow West of Castle
N=1
S=4
E=51
W=52
U=51
D=50
I=51
O=50

3—Ledge East of Castle
N=1
S=54
E=54
W=51
U=51
D=51
I=51
O=50

4—Grove South of Castle
N=51
S=54
E=54
W=2
U=51
D=5
I=53
O=50

5—Kitchen
N=6
S=4
E=9
W=6
U=50
D=13
I=9
O=6

6—Courtyard
N=7
S=5
E=9
W=12
U=7
D=5
I=9
O=7

7—Gatehouse
N=56
S=6
E=8
W=8
U=8
D=6
I=6
O=56

8—On the Walls
N=7
S=10
E=10
W=7
U=57
D=51
I=7
O=50

9—Great Hall
N=6
S=50
E=50
W=5
U=10
D=13
I=5
O=6

10—Count's Chamber
N=8
S=55
E=55
W=8
U=11
D=9
I=50
O=8

11—Tower Lookout
N=54
S=54
E=54
W=54
U=57
D=10
I=50
O=50

12—Stables
N=50
S=50
E=6
W=50
U=50
D=55
I=50
O=6

Continued on page 90

INTRODUCING ACTIVISION

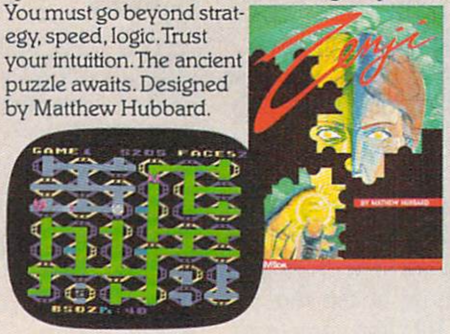
SEE YOURSELF IN A



You leave the sun behind as you lower yourself down into the unexplored caverns beneath the Peruvian jungle. Deeper and deeper you go. Past Amazon frogs, condors, and attacking bats. Across eel-infested underground rivers. From cavern to cavern, level to level. Swimming, running, dodging, stumbling, you search for the gold, the Raj diamond and the thing you really treasure…adventure. Head for it. Designed by David Crane.



You have heard the elder speak of one central source and a maze of unconnected grey paths. As you connect each grey path to the central source, what was grey becomes the green of life. When all are connected, then you have achieved "Zenji." But beware the flames and sparks of distraction that move along the paths. You must go beyond strategy, speed, logic. Trust your intuition. The ancient puzzle awaits. Designed by Matthew Hubbard.



You strap on your helicopter prop-pack, check your laser helmet and dynamite. There's no predicting what you'll have to go through to get to the trapped miners. Blocked shafts, molten lava, animals, insects, who knows what lies below. But you'll go, you're in charge of the Helicopter Emergency Rescue Operation. The miners have only one chance. You. The opening shaft is cleared now, it's time to go. Designed by John Van Ryzin.

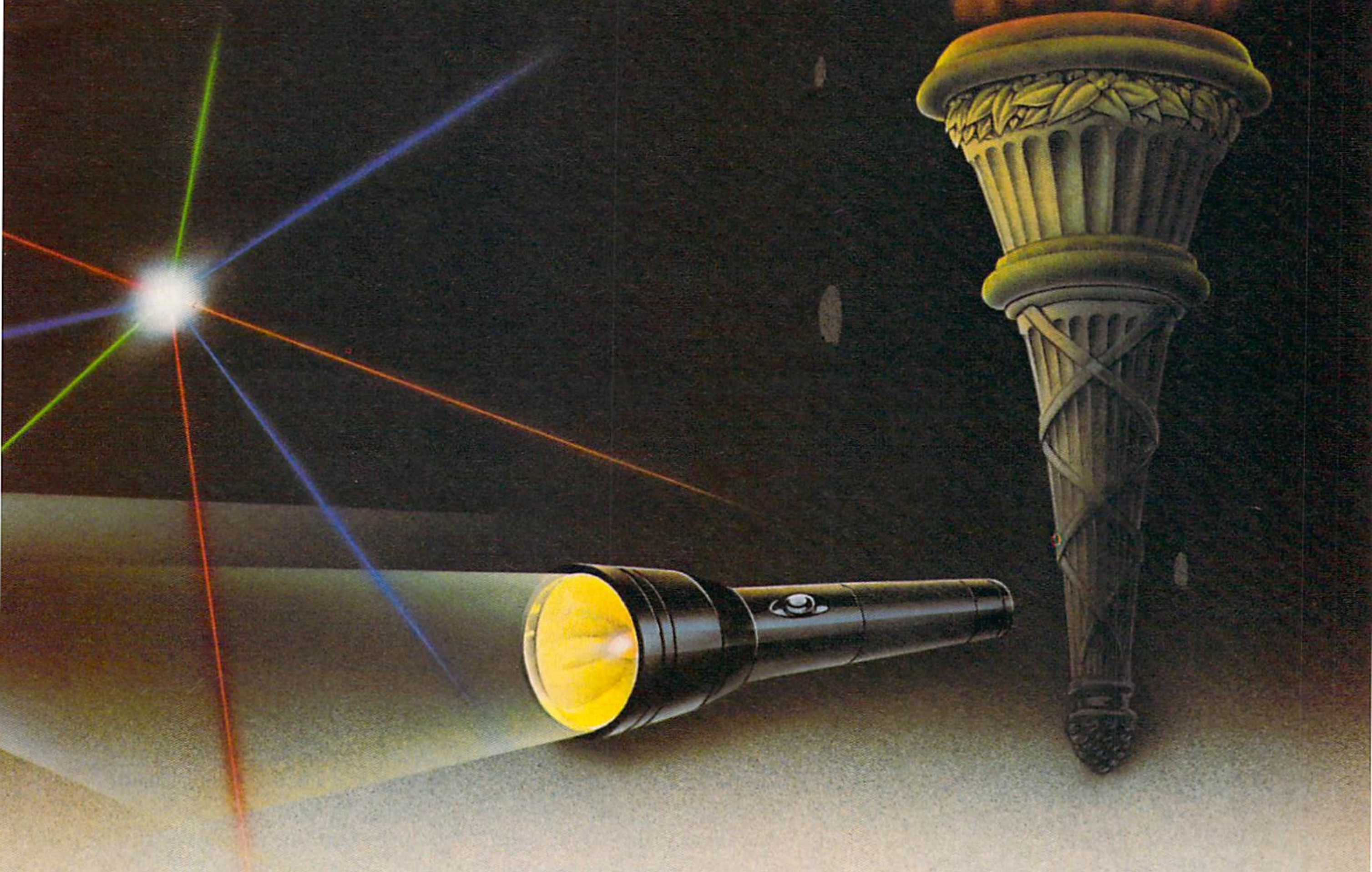




What if you were sitting in front of your Commodore 64™ programming your own Pitfall Harry™ adventure? It can happen with a little help from the creator of Pitfall Harry: David Crane. Just write your name and address on a piece of paper, tape 25¢ to it for postage and handling and mail to: The Activision C-64 Club, P.O. Box 7287, Mountain View, CA 94039. We'll send you David's Booklet, "Programming Pitfall Harry." It includes a written program that helps you create your own adventure. Go for it.

FOR YOUR COMMODORE 64.

DIFFERENT LIGHT.



As you suit up you see the webbed forcefield surrounding your planet. Holding it. Trapped with no escape. No hope. Except you: The Beamrider. The freedom of millions depends on you. Alone you speed along the grid of beams that strangle your planet. You must destroy the grid sector by sector. Your skills and your reflexes alone will determine the future of your people. Take their future in your hands. Designed by Dave Rolfe.



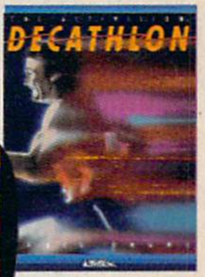


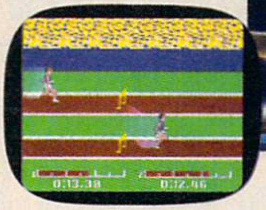
You can almost hear the quiet. And it's your job to keep it that way. A toy factory at midnight. Did you hear something? Guess not. Wrong! Suddenly balloon valves open, conveyor belts move and a whole factory full of toys goes wild. Even the robot, their latest development, is on the loose and after you. Capture the runaway toys. Restore order. Restore peace. Restore quiet. Do something! Hurry! Designed by Mark Turmell.





You made it. The Olympics. You hear languages you've never heard. And the universal roar of the crowd. You will run. Hurl. Vault. Jump. Ten events. One chance. You will push yourself this time. Further than ever. Harder than ever. But then . . . so will everyone. The competition increases, now two can compete at the same time. The crowd quiets. The starting gun sounds. A blur of adrenalin. Let the games begin. Designed by David Crane.





ACTIVISION.

We put you in the game.

Reader Service No. 226



Cowboy, *from quick-on-the-draw Wayne Schmidt and* Peripheral Vision.
READER SERVICE NO. 222

GRAPHICS PROGRAMS ON THE C–64

Continued from page 44
the offset was to reboot the entire program.

COLOR AND PERIPHERAL VISION

Peripheral Vision, as a multicolor package, allows for up to four colors in each character cell. An onscreen color menu in both normal and zoom modes assisted in the color selection process. We did find several restrictions in the way *Peripheral Vision* handled color.

Most obvious was the default selection of white as a background color with no option for change. To an extent, this would be a limitation of the light pen as an input device. A bright screen does make it easy to sense position. We did feel that this limitation could have been easily overcome by allowing for a global change of background color after completion of the image. After all, the background color is controled by only a single byte. While we are on the topic, the dark colors, in particular red and black, desensitize light pen operation. On several occasions we felt the need for a global color swap to allow better light pen operation.

Less obvious, but perhaps more significant, was the restriction on color combinations within a character cell. The color menu is divided up into three groups of five colors (the sixteenth color is the white background). Colors from the same group could not be mixed in a common character cell. Apparently each color group is assigned to one of the three nybbles which control multicolor mode in the Commodore 64. These are the two nybbles of each byte in screen memory and the single nybble of color memory. We felt that the color grouping restricted artistic expression especially with regard to subtle shade combinations. This was not a fatal limitation as can be seen from Wayne Schmidt's rendition of *Cowboy* with this program.

ZOOM MODE

The zoom mode was well done, although it did lack a simultaneous actual size image of the zoomed area. The cursor keys scrolled the zoom window to any point on the screen image. One aspect of the zoom mode was rather irritating. It allowed for the apparent use of colors in adjacent pixels which were normally forbidden by the program's color grouping. However, on exiting zoom mode, the colors would unpredictably revert to their predefined limitations. An onscreen grid, which defines the character cell positions in zoom mode, would be a big help.

GEOMETRICS

Peripheral Vision provides a complete set of geometrics, including circles, rectangles, lines (singly and end-to-end) and triangles. Circles were of the single axis round variety. Rectangles were set by independently locating the diagonal ends. Triangles required three points, of course. The menu provides for an abort of any operation before completion. A countdown of the remaining steps is maintained as well.

TEXTURES

The texture facility with *Peripheral Vision* was the best we have seen. Thirty-five predefined textures are included with the program. The user can define any number of additional textures which are stored on the disk in groups of six. Each pattern may contain two colors besides the background. These colors are the

working color and a menu selectable second color. The colors are selected at the time the texture is used. Textures may be defined only at the very start of the session as a part of the program initialization sequence. There is no way to return to texture definition once the program load is complete.

FILL MODE

The fill mode worked in either solid colors or with any of the textures. However, a continuous boundary, in a single color, is required for the fill to operate. As a result, the fill operation tended to be erratic at times. Combining fill mode with the lines feature allowed for the fastest generation of patchwork quilt patterns we have ever seen.

MISCELLANEOUS FEATURES

A selection of five brush widths, ranging from a single pixel to a broad sweep, are available. A five step "focus" or averaging option allows for manual jitter when working with the light pen. We found that a focus setting of three gave a reasonable compromise between speed and accuracy. The number four setting was quite accurate but a trifle slow. The four-quadrant, kaleidoscope, mirror feature was fun to work with. We did miss a pure horizontal or vertical mirror.

TEXT MODE

Text may be entered at any point on the display. Character selection is sequentially performed with the light pen. This could result in long selection delays while cycling through the character set. Only double size characters are available from the upper case graphics Commodore character set. Precise placement of characters was at times difficult. An automatic character positioning routine, as with *DOODLE!*, would have been a big help. The white character background obliterated underlying colors.

MOVE AND COPY MODES

Peripheral Vision provides three distinct copy modes. The first traces a single pixel outline and relocates it. The second mode is similar, but leaves the original intact allowing multiple copies. The third is a solid block copy of a filled area leaving the original in place and permitting multiple copies. The first mode was nice to have around. It allowed for the creation of a shape outline and subsequent fill with one of the textures, followed by removal of the outline. The result leaves a well-defined, borderless texture area. The shape to receive this treatment should be fairly well isolated to allow the pixel grab to function properly.

PRINTER OPTION

Peripheral Vision was the only program to include a multicolor screen dump as part of the package. We were so enthused by this feature that we spent some time converting image files from the other programs for printout by *Peripheral Vision*. This met with limited success. The main limitation seemed to be the built-in color limitations of the main program. The program always assumes a background color of white and certain allowable color combinations within a character cell. The printer routines were apparently written with these restrictions in mind. Images which substantially violated these rules were not properly translated by the printer routine.

The printer mode worked very well with the Commodore 1525 printer. Individual pixels were printed in a three horizontal by two vertical dot matrix. A different pattern was used to translate the colors into a gray scale. A full image was eight inches wide by six and one quarter inches high. The resulting 1.28 aspect ratio was very close to the 1.26 ratio of a 13 inch color monitor. The proportions with a Star Micronics Gemini 10X were not nearly as well. In this case an eight inch wide by five inch high image resulted. Apparently the program was optimized for the 1525.

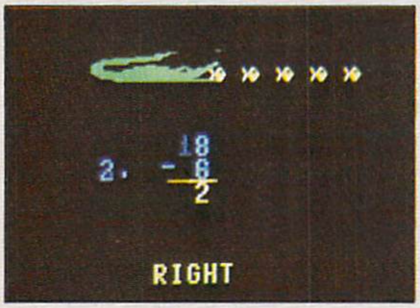
DISK OPERATIONS

The light pen driven disk mode included its own set of menus. *Peripheral Vision* files are saved in two parts, the bit map with screen memory and the color memory. The bit map file is preceded by the British pound symbol allowing the program to identify its own files. Only *Peripheral Vision* files are displayed by the automatic menu. Only LOAD and SAVE operations can be performed from the disk mode. The number of free blocks on the disk are not displayed.

CONCLUSIONS

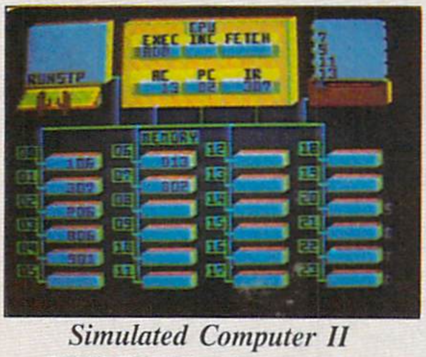
When purchased with the accompanying light pen, *Peripheral Vision* is an exceptional value. The inclusion of the multicolor screen dump routine and the powerful texture feature make the package well worthwhile. The text mode was too clumsy to allow for extensive use as a custom letterhead generator. All in all, with the inclusion of a few minor enhancements as well as a joystick option, this could be a very powerful package indeed. If you already own a light pen, then this $39.95 program is certainly worth considering. □

Educational Software: Summary



Frenzy!
READER SERVICE NO. 206

For six months, we have reviewed the general strengths and weaknesses of educational software. Many programs have been used as examples of specific traits. In summary, here is a list of the key points you might look for as you shop.



Simulated Computer II
READER SERVICE NO. 207

Information

Have you heard about this program from:

- magazine reviews?
- friends?
- teachers?

Personal Values and Goals

What topics would you like your child to study?

Does this software fit with your child's school program?

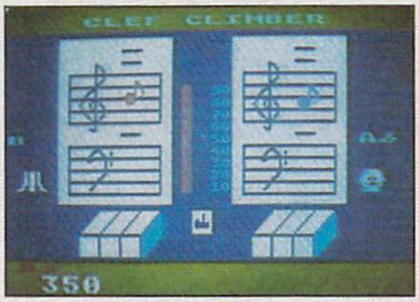
Does this program's approach offend you (violent, sexist, etc.)?

Advertising and Documentation

Has the program been well-advertised?

Does the packaging give:

- educational goals?
- age or grade levels?

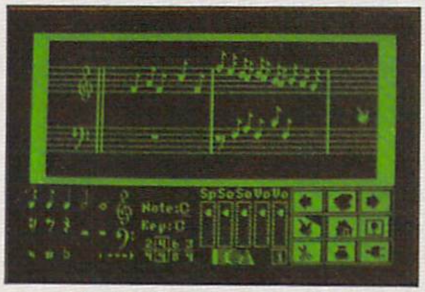


Brain Strainers
READER SERVICE NO. 208

- prerequisite skills?
- sample screen displays?

Does the documentation:

- use correct grammar and spelling?
- completely describe how to operate the program?
- make sense to the child who will use the program?
- contain supplemental information on the subject?



Music Construction Set
READER SERVICE NO. 209



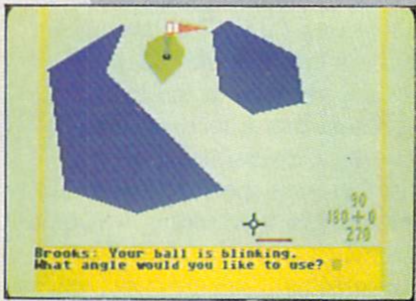
Meteor Multiplication
READER SERVICE NO. 210

Ease of Use

Is the program easy to load?

If it is a cartridge, is the program suitably implemented in the cartridge's limited memory?

Can you skip the instructions or return to them in the middle of the program?



Golf Classic
READER SERVICE NO. 211

Does the program use upper and lower case?

Do any sounds and graphics help the child to focus on the problem, or are they distracting?

Is input (by joystick, keyboard, etc.) appropriate for the child's age?

Does the program use the computer to present the material in a

A Guide for Parents

By Richard Herring

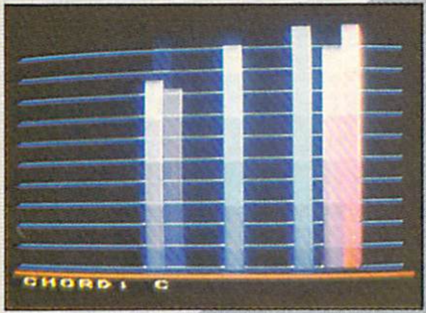
way that a workbook can't?

Error Handling and Warranty

Can you crash the program? (Can your kid?)

Does the program disable the stop and restore keys?

Does the program ignore accidental key presses (like letters when the answer must be a number)?



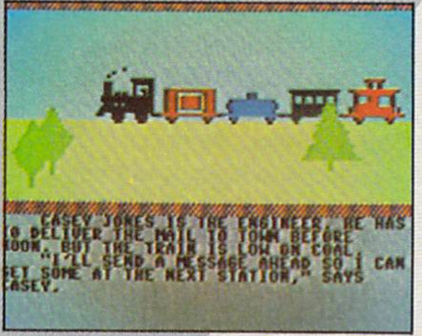
Dancing Feats
READER SERVICE NO. 212

Can you edit answers before pressing <RETURN>?

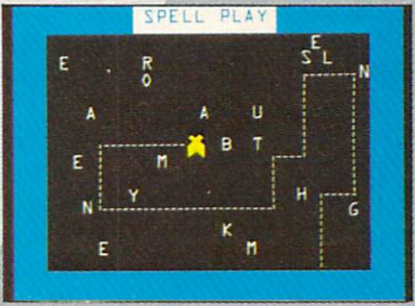
If the program is user-modifiable, are the instructions clear?

Are supplemental programs or data files available?

Does the program come with a backup or can you buy one for a reasonable price?



Kidwriter
READER SERVICE NO. 213



Spellbound
READER SERVICE NO. 214

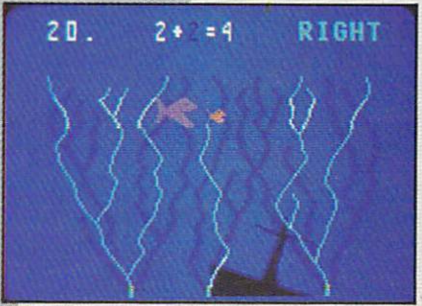
Educational Value

What are the author's background and education?

Was the program written by a team, including educators, programmers, and others?

What are the company's credentials in the education field?

How good have the company's



Gulp!
READER SERVICE NO. 215

previous products been?

Are the program's stated objectives clear when it's running?

Does the program give you some way (like scores) to evaluate your child's performance over time?

Does the program scold the child for incorrect responses?

Are the problems boring and repetitious?

Are problems presented in traditional ways that your child will recognize?

Does the program have reasonable time limits?

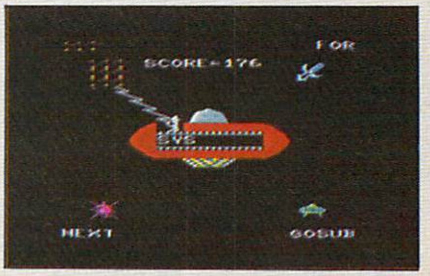
Can several children (or adults) play at once? Must they take turns, compete directly, or collaborate?



Wiz Words
READER SERVICE NO. 216

Which approach does the program take:

- drill and practice (good for memorizing math tables and spelling)?
- tutorial (where information is presented, comprehension is tested, and reviews given)?
- simulation (where hypothetical environments are explored)?
- problem solving (which allows the child to be the most creative)?



Master Type
READER SERVICE NO. 217

THE VIC 40

Continued from page 46

be available at no extra memory cost, thanks to the way reversed characters are generated.

COLOR WITH THE VIC 40

As previously mentioned, to change a color using *VIC 40* you must press CTRL and a number as is normally done. But you must also include the SHIFT key so the pause won't occur. Note that if your machine seems to lock up it is because the pause is on; to turn it off simply tap a SHIFT key. In any event, when you successfully do a cursor color change, notice that every character on the screen is changed to that color. To allow a full 8-color palette on the screen at once requires some POKEing around!

Color on the *VIC 40* is static. That is, it does not change until changed; even after a screen scroll the color remains intact. There are a total of 220 color blocks which you may set to any of the VIC's colors. Every block encompasses a 16×8 dot area, so every 4 characters on the screen may be a different color. Only those within the 16×8 (4 characters maximum) will have to be the same. On each horizontal line are 20 available color blocks. In each vertical column are 11 color blocks. To change values, or to read current ones, the POKE and PEEK statements may be used. The first color block is at 37888, with each incrementing value going horizontally. Here is a small diagram explaining that idea.

Color Line 1 37888 37889 37890 . . . 37907
Color Line 2 37908 37909 37910 . . . 37927

This pattern continues for each of the 11 horizontal lines, with color memory going across to the end of the line, then down to the next one. To see this block effect more closely, type:

```
POKE 36879,0 : POKE 37888,6
```

then hit the RETURN key. Note that the block in the upper left hand corner gets changed to blue. Move your cursor in there and type a few letters. Note how the background is independent of the characters over them (almost like color sprites). Experiment with location 36879 (the screen/border control in the VIC) and with different color block values. Some truly remarkable and graphically amazing screens are possible! (Remember that the color blocks are static. In case you scroll down and the characters move up, notice that the color does not! This is very useful when preparing displays for any purpose.)

SCREEN RAM WITH THE VIC 40

Storing characters via POKE works in the same manner as with all Commodore computers. You use the method

```
POKE SCREEN LOCATION, CHARACTER V
ALUE
```

where screen location is some number on the screen RAM, and character value is a number from 0 to 255 corresponding to a certain character (see your user guide or programmer's reference manual for a list of the codes). The big difference with *VIC 40*'s screen RAM is that it matches exactly the PET/CBM's screen RAM. This enables you to run BASIC software written for your PET/CBM on the VIC! The beginning location for screen RAM is 32768 and it continues up to and including 33767. That corresponds to 1000 characters onscreen at once, exactly what the 40 by 25 screen generates.

To store a character at the top left part of the screen, type

```
POKE 32768,1
```

and hit RETURN. The letter 'A' should appear at the HOME position of your screen. This isn't meant to be a tutorial on how to use the POKE command, so if you don't fully understand its use, please consult your user guide. Otherwise, happy PEEKing and POKEing!

HI-RES SCREEN ON THE VIC 40

VIC 40's screen is not a typical screen in the sense that it is not stored as characters, but rather as bits. *VIC 40*'s screen is a bit-mapped hi-res page in RAM. *VIC 40* gives you the ability to POKE characters onto the screen as with a standard PET. Why use hi-res then? Simply because you can now control every dot on the screen, and therefore create superb hi-res pictures combining both dots and text.

The hi-res screen in *VIC 40* measures 160 dots across by 176 dots vertically. The VIC 20 allows larger than normal characters (16×8, instead of 8×8) on its graphics chip, and this is the mode which allows *VIC 40* to allow such a large display. By placing these 16×8 characters one on top of another in rows in RAM, and by repeatedly changing their values for each different character displayed, the *VIC 40* can simulate a 40×25 machine. To allow quicker scrolls and altogether smoother operation, the programmable characters are stored vertically (as said before) so simple byte moves suffice when a scroll takes place. (Older methods involve storing characters side by side in rows, which forces you to do bit-shifts in order to move the screen up or down! This method is better suited for sideways scrolling [as in games].) Notice that the scroll does slow *VIC 40* down a bit, as it has to move all the hi-res lines up

8 bytes, but the speed required for the older method would have taken 3 times as long!

If the previous paragraph confused you, take heed, this one is for you. Whereas the previous info was for more advanced graphics users, the data given here will be of use to all. Now you can learn how to plot on the hi-res screen, with characters also there!

Remember that the *VIC 40* screen is only a hi-res page 160 dots across by 176 dots vertically. By using the following formula you can actually light single dots (pixels) on the screen. You have a large selection as there are 176×160 points to choose from. More advanced people will be able to write routines to turn points off, to draw lines, and to actually paint in certain polygons. I leave that up to you, but in line 5000 of the *VIC 40 Demo* is a general purpose plot subroutine which the demo uses to generate pixels (picture elements) on the screen. Feel free to use it in your own *VIC 40* programs, and try to use the knowledge given so far to further understand its operations.

Type in, SAVE, and RUN the demo for the *VIC 40.* Notice how neat color displays can be created easily, how both character sets can be used onscreen at once, and how points can be plotted (overlayed) onto the *VIC 40* screen. Notice that I say overlay because the points plotted can be over the entire screen, yet the characters underneath them may be recalled in a flash (even if your plot routine erases them). To see what I mean, break out of the *Demo* during the sine wave generation. Move your cursor over a character with points plotted over it. See what happens? The points plotted will disappear and the character previously underneath it will reappear intact. This is another power of *VIC 40*; a whole hi-res screen can be plotted, but underneath it can be a screen full of CBM characters which can be recalled instantly. This is another case of pseudo-sprite ability, which emulates the foreground-background ability of the Commodore 64 sprites.

As always, the key to getting nice display screens is *experimentation*! A word processor with graphic charts, or a database combining all elements that a businessman could desire (similar to *LOTUS 1-2-3*), or even a graphics tablet for game design is entirely possible with *VIC 40*. With 8K RAM expansion (or even 32K), the software possible for this operating system can rival that of any Commodore computer, 64 included! Don't fret, VIC users, this package will keep you and your initial computer investment busy for a long, long time, even though CBM no longer manufactures VIC 20's.

USING ML WITH THE VIC 40

For any ultimately serious and marketable product, machine language is a key. To execute number-crunching quick sorts, or fast I/O, ML is the answer. If you do not yet know machine language, please skip this area and go on to the SOUND section.

The *VIC 40* changes the IRQ vector to its own special input and output handlers. All input and output called by the kernal first goes through the *VIC 40 Operating System* routines before branching to the standard Commodore ones. The reason for this is obvious. Since all routines are character-oriented, and formatted for 22 columns, the ROM routines would never work in 40 columns. By changing vectors at the beginning of page 3 ($300-$330), *VIC 40* is wedged into the standard operating system.

The *VIC 40* is very intelligent with regard to interrupts in that it will allow IRQ's besides its own to occur. In other words, you may want to have a special keyboard reader, or perhaps a music maker, use the normal 60 times a second IRQ interrupt besides the *VIC 40*. *VIC 40* can handle it with ease by modifying itself and jumping to the old interrupt when finished. This 'chaining' of interrupts can lead to *very* powerful structures, all occurring 60 times a second.

The NMI (warm start interrupt) is checked the same way as the IRQ. If one different from the norm is present when the *VIC 40* is executed, it will prevail over *VIC 40*'s. This is useful if you have some business software which uses the NMI to reset itself or to trap errors. The normal NMI of the *VIC 40* sends the *VIC* back into 22-column mode with all program lines and variables left intact.

Note that to initiate a warm start, the STOP key need not be pressed along with the RESTORE key. Since this NMI is non-destructive to memory (variables and program lines) it eases its use. Of course, you can change the NMI to meet your needs, which may mean disabling it entirely!

The ML portion of the *VIC 40* devotes itself to maintaining the bitmap and executing time-consuming tasks such as maintaining the screen editor's line links and executing character (bit by bit) scrolls. Another feature of *VIC 40* which requires ML intervention is the acceptance of PET/CBM POKE's. Even though there is no RAM at 32768 to 33767, the *VIC* is tricked into thinking that there is. This requires a test of all BASIC statements as they are executed, and patches to change PET/CBM POKE's to ones which can be used on the *VIC*. See the memory map for "real" locations of the character screen. The upper/lower case changer along with the sound POKE's are handled the same way. A little interception and a bit of gritty programming can fool BASIC into POKEing and PEEKing RAM which doesn't exist and never will.

SOUND ON THE VIC 40

Sound on the *VIC 40* maintains the same characteristics as the VIC 20 possesses. A few additions are made, though. PET/CBM sounds will now work

Continued on page 89

PROGRAM LISTINGS

On the following pages are listed several programs that we hope you'll want to punch in your Commodore computer. But please read the following introduction first; there are a few things you'll need to know.

Certain computer commands are displayed on the monitor by a variety of odd-looking characters. To get your computer to display these commands rather than actually perform them, you'll need to enter the quote mode. Hold down the SHIFT key and press the "2" key; a set of quote marks will appear. This tells the computer that the characters that follow are to be displayed, not performed. To exit the quote mode, type another set of quote marks, or hit the RETURN key. You'll also enter the quote mode when you INserT spaces or characters onto a line.

In *Ahoy!'s* program listings, you'll frequently find letters and/or numbers surrounded by brackets { }. That's because, for the purposes of clear reproduction, we at *Ahoy!* use a daisy wheel printer incapable of reproducing command symbols. For example, when you're in the quote mode and press the SHIFT and CLR/HOME keys at the same time, the screen (or a dot-matrix printer) will indicate this command with a heart { }. Because a daisy wheel cannot duplicate this symbol, it substitutes an alternate code between brackets. In the case of the SHIFT/CLR HOME symbol, our printer substitutes {SC}.

Another special case is SHIFT and COMMODORE characters. We represent these by underlining or overlining, respectively: any character underlined in the program listing should be punched in as a SHIFTed character (J̲ = SHIFT J), any character overlined should be punched in as a COMMODORE character (J̄ = COMMODORE J).

An alternate way of entering commands and other graphics symbols and characters is to use their corresponding character strings. The CLR/HOME command, for example, is entered by typing CHR$(147). While this requires a few extra strokes, it facilitates editing your program or reading the printed listing. For a complete list of CHR$ codes, consult the appendix at the back of your Commodore user manual.

Below is a list of the command abbreviations you'll find in our program listings, the commands they stand for, how to enter them, and how they'll appear on the screen or on a dot matrix printout.

When You See	It Means	You Type	You Will See	When You See	It Means	You Type	You Will See
{SC}	Screen Clear	SHIFT CLR/HOME		{YL}	Yellow	CNTRL 8	
{HM}	Home	CLR/HOME		{OR}	Orange	COMMODORE 1	
{CU}	Cursor Up	SHIFT ↑ CRSR ↓		{BR}	Brown	COMMODORE 2	
{CD}	Cursor Down	↑ CRSR ↓		{LR}	Light Red	COMMODORE 3	
{CL}	Cursor Left	SHIFT ← CRSR →		{G1}	Grey 1	COMMODORE 4	
{CR}	Cursor Right	← CRSR →		{G2}	Grey 2	COMMODORE 5	
{SS}	Shifted Space	SHIFT space		{LG}	Light Green	COMMODORE 6	
{IN}	Insert	INST		{LB}	Light Blue	COMMODORE 7	
{RV}	Reverse On	CNTRL 9		{G3}	Grey 3	COMMODORE 8	
{RO}	Reverse Off	CNTRL 0		{F1}	Function 1	F 1	
{BK}	Black	CNTRL 1		{F2}	Function 2	F 2	
{WH}	White	CNTRL 2		{F3}	Function 3	F 3	
{RD}	Red	CNTRL 3		{F4}	Function 4	F 4	
{CY}	Cyan	CNTRL 4		{F5}	Function 5	F 5	
{PU}	Purple	CNTRL 5		{F6}	Function 6	F 6	
{GN}	Green	CNTRL 6		{F7}	Function 7	F 7	
{BL}	Blue	CNTRL 7		{F8}	Function 8	F 8	

IMPORTANT!
Before typing in the *Bug Repellent* and other *Ahoy!* programs, refer to the information on page 67.

VIC 20 BUG REPELLENT

By Michael Kleinert and David Barron

The program listed below will allow you to quickly debug any *Ahoy!* program you type in on your VIC 20. Follow directions for cassette or disk.

For cassette: type in and save the *Bug Repellent* program, then type RUN 63000[RETURN]SYS 828[RETURN]. If you typed the program properly, it will generate a set of two-letter line codes that will match those listed below the program on this page. (If you didn't type the program properly, of course, no line codes will be generated. You'll have to debug the *Bug Repellent* itself the hard way.)

Once you've got a working *Bug Repellent*, type in the program you wish to check. Save it and type the RUN and SYS commands listed above once again, then compare the line codes generated to those listed in the magazine. If you spot a discrepancy, a typing error exists in that line. Important: you must use exactly the same spacing as the program in the magazine. Due to memory limitations on the VIC, the VIC *Bug Repellent* will register an error if your spacing varies from what's printed.

You may type SYS 828 as many times as you wish, but if you use the cassette for anything, type RUN 63000 to restore the *Repellent*.

When your program has been disinfected you may delete all lines from 63000 on. (Be sure the program you type doesn't include lines above 63000!)

For disk: type in the *Bug Repellent*, save it, and type RUN:NEW[RETURN]. (See above regarding testing the *Bug Repellent* on itself.) Type in the program you wish to check, then SYS 828. This will generate a set of two-letter line codes that you should compare to those listed in the magazine.

To pause the line codes listing, press SHIFT. To permanently pause it, press SHIFT LOCK. To continue, release SHIFT LOCK.

To send the list to the printer type OPEN 4,4:CMD 4:SYS 828[RETURN]. When the cursor comes back, type PRINT#4:CLOSE 4[RETURN].

```
•63000 FOR X = 828 TO 1023 :READ Y
 :POKE X,Y:NEXT:END
•63001 DATA 169, 0, 133, 63, 133,
 64, 165, 43, 133, 251
•63002 DATA 165, 44, 133, 252, 160
 , 0, 132, 254, 32, 228
•63003 DATA 3, 234, 177, 251, 208,
 3, 76, 208, 3, 230
•63004 DATA 251, 208, 2, 230, 252,
 169, 244, 160, 3, 32
•63005 DATA 30, 203, 160, 0, 177,
 251, 170, 230, 251, 208
•63006 DATA 2, 230, 252, 177, 251,
 32, 205, 221, 169, 58
•63007 DATA 32, 210, 255, 169, 0,
 133, 253, 230, 254, 32
•63008 DATA 228, 3, 234, 165, 253,
 160, 0, 170, 177, 251
•63009 DATA 201, 32, 240, 6, 138,
 113, 251, 69, 254, 170
•63010 DATA 138, 133, 253, 177, 25
 1, 208, 226, 165, 253, 41
•63011 DATA 240, 74, 74, 74, 74, 2
 4, 105, 65, 32, 210
•63012 DATA 255, 165, 253, 41, 15,
 24, 105, 65, 32, 210
•63013 DATA 255, 169, 13, 32, 210,
 255, 173, 141, 2, 41
•63014 DATA 1, 208, 249, 230, 63,
 208, 2, 230, 64, 230
•63015 DATA 251, 208, 2, 230, 252,
 76, 74, 3, 169, 236
•63016 DATA 160, 3, 32, 30, 203, 1
 66, 63, 165, 64, 32
•63017 DATA 205, 221, 169, 13, 32,
 210, 255, 96, 230, 251
•63018 DATA 208, 2, 230, 252, 96,
 0, 76, 73, 78, 69
•63019 DATA 83, 58, 32, 0, 76, 73,
 78, 69, 32, 35
•63020 DATA 32, 0, 0, 0, 0, 0
```

BUG REPELLENT LINE CODES FOR VIC 20 BUG REPELLENT

LINE # 63000:MH	LINE # 63011:NN
LINE # 63001:BD	LINE # 63012:IG
LINE # 63002:FO	LINE # 63013:EN
LINE # 63003:ND	LINE # 63014:GJ
LINE # 63004:DJ	LINE # 63015:IK
LINE # 63005:LP	LINE # 63016:HG
LINE # 63006:JB	LINE # 63017:CK
LINE # 63007:JF	LINE # 63018:JF
LINE # 63008:KA	LINE # 63019:OH
LINE # 63009:HP	LINE # 63020:LH
LINE # 63010:KJ	LINES: 21

C-64 BUG REPELLENT

By Michael Kleinert and David Barron

The program listed below will allow you to quickly debug any *Ahoy!* program you type in on your C-64.

Type in, SAVE, and RUN the *Bug Repellent,* Type NEW, then type in or LOAD the *Ahoy!* program you wish to check. When that's done, SAVE your program (don't RUN it!) and type SYS 49152 [RETURN]. You'll be asked if you want the line value codes displayed on the screen or dumped to the printer. If you select screen, it will appear there.

The table will move quickly, too quickly for most mortals to follow. To pause the listing depress and hold the SHIFT key. To pause for an extended period, depress SHIFT LOCK. As long as it is locked, the display will remain frozen.

Compare the table your machine generates to the table in *Ahoy!* that follows the program you're entering: If you spot a difference, an error exists in that line. Jot down the numbers of lines where contradictions occur, LIST each line, spot the errors, and correct them.

```
•5000 FOR X = 49152 TO 49488 :READ
 Y:POKE X,Y:NEXT:END
•5001 DATA 32, 161, 192, 165, 43,
 133, 251, 165, 44, 133
•5002 DATA 252, 160, 0, 132, 254,
 32, 37, 193, 234, 177
•5003 DATA 251, 208, 3, 76, 138, 1
92, 230, 251, 208, 2
•5004 DATA 230, 252, 76, 43, 192,
76, 73, 78, 69, 32
•5005 DATA 35, 32, 0, 169, 35, 160
, 192, 32, 30, 171
•5006 DATA 160, 0, 177, 251, 170,
230, 251, 208, 2, 230
•5007 DATA 252, 177, 251, 32, 205,
 189, 169, 58, 32, 210
•5008 DATA 255, 169, 0, 133, 253,
230, 254, 32, 37, 193
•5009 DATA 234, 165, 253, 160, 0,
76, 13, 193, 133, 253
•5010 DATA 177, 251, 208, 237, 165
, 253, 41, 240, 74, 74
•5011 DATA 74, 74, 24, 105, 65, 32
, 210, 255, 165, 253
•5012 DATA 41, 15, 24, 105, 65, 32
, 210, 255, 169, 13
•5013 DATA 32, 220, 192, 230, 63,
208, 2, 230, 64, 230
•5014 DATA 251, 208, 2, 230, 252,
76, 11, 192, 169, 153
•5015 DATA 160, 192, 32, 30, 171,
166, 63, 165, 64, 76
•5016 DATA 231, 192, 96, 76, 73, 7
8, 69, 83, 58, 32
•5017 DATA 0, 169, 247, 160, 192,
32, 30, 171, 169, 3
•5018 DATA 133, 254, 32, 228, 255,
 201, 83, 240, 6, 201
•5019 DATA 80, 208, 245, 230, 254,
 32, 210, 255, 169, 4
•5020 DATA 166, 254, 160, 255, 32,
 186, 255, 169, 0, 133
•5021 DATA 63, 133, 64, 133, 2, 32
, 189, 255, 32, 192
•5022 DATA 255, 166, 254, 32, 201,
 255, 76, 73, 193, 96
•5023 DATA 32, 210, 255, 173, 141,
 2, 41, 1, 208, 249
•5024 DATA 96, 32, 205, 189, 169,
13, 32, 210, 255, 32
•5025 DATA 204, 255, 169, 4, 76, 1
95, 255, 147, 83, 67
•5026 DATA 82, 69, 69, 78, 32, 79,
 82, 32, 80, 82
•5027 DATA 73, 78, 84, 69, 82, 32,
 63, 32, 0, 76
•5028 DATA 44, 193, 234, 177, 251,
 201, 32, 240, 6, 138
•5029 DATA 113, 251, 69, 254, 170,
 138, 76, 88, 192, 0
•5030 DATA 0, 0, 0, 230, 251, 208,
 2, 230, 252, 96
•5031 DATA 170, 177, 251, 201, 34,
 208, 6, 165, 2, 73
•5032 DATA 255, 133, 2, 165, 2, 20
8, 218, 177, 251, 201
•5033 DATA 32, 208, 212, 198, 254,
 76, 29, 193, 0, 169
•5034 DATA 13, 76, 210, 255, 0, 0,
 0
```

BUG REPELLENT LINE CODES FOR C-64 BUG REPELLENT

LINE # 5000:GJ	LINE # 5018:FK
LINE # 5001:DL	LINE # 5019:FL
LINE # 5002:DB	LINE # 5020:CL
LINE # 5003:OF	LINE # 5021:GC
LINE # 5004:KN	LINE # 5022:NN
LINE # 5005:CA	LINE # 5023:NH
LINE # 5006:CE	LINE # 5024:IM
LINE # 5007:JE	LINE # 5025:KC
LINE # 5008:CL	LINE # 5026:DC
LINE # 5009:NB	LINE # 5027:ML
LINE # 5010:MB	LINE # 5028:GN
LINE # 5011:EP	LINE # 5029:JK
LINE # 5012:GH	LINE # 5030:NA
LINE # 5013:AN	LINE # 5031:DM
LINE # 5014:NG	LINE # 5032:JA
LINE # 5015:BF	LINE # 5033:FM
LINE # 5016:EP	LINE # 5034:PA
LINE # 5017:PJ	LINES: 35

EMERALD ELEPHANT

FROM PAGE 95

```
•1 REM "THE EMERALD ELEPHANT OF CI
 PANGU"
•2 REM A MINI-ADVENTURE, JUST TO S
 HOW HOW IT'S DONE.  FOR THE 64--I
 T WON'T FIT
•3 REM ON AN UNEXPANDED VIC.  IF Y
 OU HAVE AN EXPANDED VIC, CHANGE T
 HE TEXT SO IT
•4 REM WILL FIT PROPERLY ON THE SC
 REEN.
•5 REM
•6 REM RD=ROOM DIRECTION TABLE.  C
 W$=COMMAND WORD TABLE.  RN$=ROOM
 NAMES
•7 REM KS=KEYSTROKE TABLE.  TN$="T
 HING" NAME TABLE.  TS$="THING" SH
 ORT NAMES
•8 REM TL="THING" LOCATION TABLE (
 ROOM #).  VS=VISIT TABLE: NON-ZER
 O IF VISITED.
•9 REM
•10 DIM RD(14,8),CW$(16),RN$(60),K
 S(64),TN$(9),TS$(9),TL(9),VS(14)
•20 GOSUB 1000
•25 GOSUB 970
•27 REM
•28 REM SET UP BLANK LINES AND POS
 ITION STRINGS
•29 REM
•30 BL$="{HM}                     {H
 M}"
•31 RL$="{HM}{CD}{CD}":FOR I=1 TO
 78:RL$=RL$+" ":NEXT:RL$=RL$+"{HM}
 {CD}{CD}"
•32 TL$="{HM}{CD}{CD}{CD}{CD}":FOR
  I=1 TO 78:TL$=TL$+" ":NEXT:TL$=T
 L$+"{HM}{CD}{CD}{CD}{CD}"
•33 DL$="{HM}{CD}{CD}{CD}{CD}{CD}{
 CD}{CD}"
•34 T$="{HM}{CD}{CD}{CD}   WITHIN
 YOUR REACH IS . . . ."
•35 C$=CHR$(13)
•60 PRINT "{SC}CAREFUL!"C$C$"JUST
 THROUGH THOSE TREES IS THE CASTLE
 "
•61 PRINT "OF DARKNESS."C$C$"ONCE
 YOU SEE IT, YOU'LL BE STRUCK BLIN
 D"
•62 PRINT "UNTIL YOU LEAVE."C$C$"B
 UT HIDDEN IN A SECRET TREASURE RO
 OM"
•63 PRINT "SOMEWHERE IN THE CASTLE
  IS THE EMERALD"
•64 PRINT "ELEPHANT OF CIPANGU.  T
 HE TOUCH OF THE"
•65 PRINT "ELEPHANT WILL GIVE YOU
 THE PRICELESS"
•66 PRINT "GIFT OF FAR SIGHT, WHIC
 H WILL LET YOU"
•67 PRINT "SEE TO THE ENDS OF THE
 EARTH.  WHATEVER"
•68 PRINT "PLACE YOU THINK OF, YOU
  WILL SEE."C$C$"(PRESS ANY KEY)":
 GOSUB 990
•69 PRINT "{SC}AS FOR ME, I SEEM T
 O YOU TO BE MERELY A"
•70 PRINT "FIREFLY.  I AM REALLY S
 OMETHING ELSE"
•71 PRINT "ENTIRELY, AND ONCE YOU
 GET THE ELEPHANT"
•72 PRINT "AND LEAVE THE CASTLE, Y
 OU WILL SEE ME ASI REALLY AM."C$
•73 PRINT "WILL YOU COME WITH ME?
  I'LL BE YOUR"
•74 PRINT "EYES, WHILE YOU DECIDE
 WHERE WE SHOULD  GO."C$
•75 PRINT "YOU CAN PRESS A SINGLE
 KEY TO GO {WH}U{LB}P,"
•76 PRINT "{WH}D{LB}OWN, {WH}I{LB}
 N, {WH}O{LB}UT, {WH}N{LB}ORTH, {W
 H}S{LB}OUTH, {WH}E{LB}AST, {WH}W{
 LB}EST, OR {WH}B{LB}ACK."C$
•77 PRINT "YOU CAN ALSO {WH}T{LB}A
 KE SOMETHING, {WH}L{LB}EAVE IT"
•78 PRINT "BEHIND, TELL ME TO {WH}
 P{LB}EER CLOSELY AT"
•79 PRINT "THINGS AROUND US, CHECK
  TO SEE WHAT WE"
•80 PRINT "HAVE {WH}G{LB}OT, {WH}Q
 {LB}UIT, OR ASK FOR {WH}H{LB}ELP
 ({WH}?{LB})."C$C$"(PRESS ANY KEY)
 "
•81 GOSUB 990
•87 REM
•88 REM INITIAL POSITION
•89 REM
•90 XX=1:XR=1:PR=1:GOSUB 350
•97 REM
•98 REM MAIN LOOP.  GET KEYSTROKE
 FIRST
•99 REM
```

```
•100 GOSUB 990
•105 PRINT BL$CW$(CM):IF CM=0 THEN
 100
•107 REM
•108 REM DIRECTION COMMANDS
•109 REM
•110 IF CM<9 THEN GOSUB 300:GOTO 1
30
•117 REM
•118 REM ALL OTHER COMMANDS
•119 REM
•120 C=CM-8:ON C GOSUB 390,400,450
,550,470,190,500,500
•127 REM
•128 REM CHECK FOR VICTORY
•129 REM
•130 IF PR=1 AND TL(9)=0 THEN FOR
I=0 TO 999:NEXT:GOTO 940
•140 PRINT BL$CW$(CM):GOTO 100
•187 REM
•188 REM "QUIT" ROUTINE
•189 REM
•190 PRINT "{SC}PRESS Q TO QUIT":G
OSUB 990:IF CM=14 THEN END
•195 GOTO 350
•297 REM
•298 REM MOVEMENT HANDLER
•299 REM
•300 XX=XR:XR=PR:PR=RD(XR,CM)
•302 REM
•303 REM CHECK FOR SPECIAL CASES
•304 REM
•310 IF XR=7 AND TL(0)=0 AND PR=56
 THEN GOSUB 900:PR=1:GOTO 350
315 IF PR=14 THEN GOSUB 905:GOTO
•350
•320 IF PR=56 AND XR=4 AND TL(2)=0
 THEN GOSUB 920:PR=5:GOTO 350
•325 IF PR=4 AND XR=5 AND TL(9)=0
THEN GOSUB 925:GOTO 350
•342 REM
343 REM IS IT AN ILLEGAL MOVEMENT
?
•344 REM
•345 DS$="":IF PR>49 THEN DS$=RN$(
PR):PR=XR:XR=XX
•346 REM
•347 REM      * * *  RETURN POINT
 * * *
•348 REM TABLE SEARCH--ARE THERE A
NY "THINGS" IN THE ROOM?
•349 REM
•350 TN$="":FOR I=0 TO 9:IF TL(I)=
PR THEN TN$=TN$+TN$(I)+", "
•355 NEXT:L=LEN(TN$):IF L>1 THEN T
N$=LEFT$(TN$,L-2)
•360 TT$=T$:IF TN$="" THEN TT$=""
•362 REM
•363 REM SET ROOM NAME TO PRESENT
ROOM
•364 REM
•365 RN$=RN$(PR)
•367 REM
•368 REM IS IT THE FIRST VISIT TO
THE ROOM?
•369 REM
•370 IF VS(PR)=0 THEN VS(PR)=1:GOT
O 550
•372 REM
•373 REM CLEAR SCREEN AND PRINT RO
OM NAME AND MESSAGE
•374 REM
•375 PRINT "{SC}"CW$(CM)RL$RN$TT$T
L$TN$DL$DS$
•380 RETURN
•387 REM
•388 REM BACK COMMAND HANDLER
•389 REM
•390 XX=PR:PR=XR:XR=XX:RN$=RN$(PR)
:DS$="":GOTO 350
•397 REM
•398 REM "TAKE" ROUTINE
•399 REM
•400 I=0
•405 IF TL(I)=PR THEN TL(I)=0:PP$=
PP$+CHR$(I):DS$="GOT IT!":GOTO 35
0
•410 IF LEN(PP$)>5 THEN GOSUB 450
•415 I=I+1:IF I>9 THEN DS$="THERE'
S NOTHING HERE.":GOTO 350
•420 GOTO 405
•447 REM
•448 REM "LEAVE" ROUTINE
•449 REM
•450 IF LEN(PP$)<1 THEN DS$="YOU D
ON'T HAVE ANYTHING NOW.":GOTO 350
•455 LT=ASC(LEFT$(PP$,1)):PP$=RIGH
T$(PP$,LEN(PP$)-1)
•460 TL(LT)=PR:DS$="":GOTO 350
•467 REM
•468 REM "GOT" HANDLER
•469 REM
•470 IF PP$="" THEN DS$="YOU'VE GO
T NOTHING RIGHT NOW.":GOTO 350
•475 DS$="YOU'VE GOT:"+C$+C$
•480 FOR I=1 TO LEN(PP$):DS$=DS$+T
```

```
S$(ASC(MID$(PP$,I,1)))+C$:NEXT
•485 GOTO 350
•497 REM
•498 REM "HELP" ROUTINE
•499 REM
•500 PRINT "{SC}"C$C$"THERE ARE 9
 POSSIBLE DIRECTIONS:"
•501 PRINT "N = NORTH"C$"S = SOUTH
 "C$"E = EAST"C$"W = WEST"C$"U = U
 P"
•502 PRINT "D = DOWN"C$"I = IN"C$"
 O = OUT"C$"B = BACK TO PREVIOUS R
 OOM"C$
•503 PRINT "OTHER COMMANDS ARE:"C$
 C$"T = TAKE THE FIRST OBJECT LIST
 ED"
•504 PRINT "L = LEAVE BEHIND WHATE
 VER YOU'VE BEEN        CARRYING LO
 NGEST"
•505 PRINT "P = PEER AT SURROUNDIN
 GS"C$"G = LIST WHAT WE'VE GOT WIT
 H US"
•506 PRINT "Q = QUIT"C$"H = HELP"C
 $"? = HELP"C$
•507 PRINT "(PRESS ANY KEY)";:GOSU
 B 990:PRINT "{SC}":GOTO 350
•547 REM
•548 REM "PEER" JUMP TABLE
•549 REM
•550 PRINT "{SC}"DL$
•555 ON PR GOSUB 600,620,640,660,6
 80,700,720,740,760,780,800,820,84
 0,860
•560 PRINT BL$CW$(CM)RL$RN$TT$TL$T
 N$
•565 RETURN
•597 REM
•598 REM MAIN GATE DESCRIPTION
•599 REM
•600 PRINT "WE ARE STANDING ON THE
  BRIDGE LEADING"
•601 PRINT "TO THE HUGE WOODEN GAT
 E OF THE CASTLE."C$
•602 PRINT "TO THE NORTH IS THE DE
 EP FOREST.  A"
•603 PRINT "PATH LEADS AROUND THE
 CASTLE TO THE"
•604 PRINT "EAST AND WEST.  THE WA
 LLS ARE VERY HIGH"
•605 PRINT "AND I CAN'T SEE ANYONE
  AT ALL."
•615 RETURN
•617 REM
•618 REM WEST OF CASTLE
•619 REM
•620 PRINT "THE MEADOW IS FULL OF
 RABBITS, SO BE"
•621 PRINT "CAREFUL WHERE YOU STEP
 .  THE FOREST IS"
•622 PRINT "THICK TO THE WEST.  A
 PATH LEADS AROUND"
•623 PRINT "THE CASTLE TO THE NORT
 H AND SOUTH."C$
•624 PRINT "THE STONES OF THE CAST
 LE WALL ARE SLICK"
•625 PRINT "AND COVERED WITH MOSS.
 "C$
•626 PRINT "LISTEN--YOU CAN HEAR A
  HORSE WHINNYING."
•635 RETURN
•637 REM
•638 REM EAST OF CASTLE
•639 REM
•640 PRINT "PLEASE DON'T DANCE ARO
 UND SO MUCH!"C$
•641 PRINT "WE'RE ON A NARROW LEDG
 E ALONG THE EAST"
•642 PRINT "EDGE OF THE CASTLE.  I
 T ENDS JUST A FEW"
•643 PRINT "YARDS TO THE SOUTH."C$
•644 PRINT "OVER THE EDGE IT'S A S
 HEER DROP FOR AT"
•645 PRINT "LEAST TWO THOUSAND FEE
 T.  THE EDGE IS"
•646 PRINT "CRUMBLING A LITTLE, SO
  IF YOU DON'T"
•647 PRINT "MIND, COULDN'T WE JUST
  GO BACK?"
•655 RETURN
•657 REM
•658 REM SOUTH OF CASTLE
•659 REM
•660 PRINT "WE'RE IN A GROVE OF TR
 EES ON A NARROW"
•661 PRINT "WEDGE OF LAND JUST SOU
 TH OF THE CASTLE."C$
•662 PRINT "THE LAND SLOPES DOWN T
 OWARD THE FOUNDA-"
•663 PRINT "TION OF THE CASTLE TO
 THE NORTH, AND TO"
•664 PRINT "THE SOUTH IT SLOPES UP
  TO A SUDDEN"
•665 PRINT "DROP-OFF.  THERE'S A S
 TEEP CLIFF TO THE EAST, TOO."C$
•666 PRINT "THERE'S A COLUMN OF SM
 OKE RISING ABOVE  THE CASTLE WALL
```

```
."
•675 RETURN
•677 REM
•678 REM KITCHEN
•679 REM
•680 PRINT "THE KITCHEN FIRE IS BU
 RNING HOTLY, BUT"
•681 PRINT "THERE'S NO ONE IN THE
 ROOM."C$
•682 PRINT "A LARGE OPEN DOORWAY L
 EADS OUT INTO THE"
•683 PRINT "COURTYARD TO THE NORTH
  AND WEST.  THE"
•684 PRINT "DOORWAY IS FILLED WITH
  FLIES.  NOT"
•685 PRINT "FIREFLIES--THE LOWER-C
 LASS KIND."C$
•686 PRINT "TO THE EAST THERE'S AN
 OTHER DOOR, AND"
•687 PRINT "TO THE SOUTH THERE'S A
  GAP WHERE ONE"
•688 PRINT "STONE HAS BEEN PULLED
 OUT OF THE"
•689 PRINT "FOUNDATION.  IT'S THE
 SECRET POSTERN"
•690 PRINT "DOOR, THE ONE WE CRAWL
 ED THROUGH."C$
•691 PRINT "THERE'S A LARGE TABLE
 IN THE MIDDLE OF"
•692 PRINT "THE ROOM, WITH MORE FO
 OD ON IT THAN A"
•693 PRINT "NORMAL PERSON COULD EA
 T IN A WEEK.";
•695 RETURN
•697 REM
•698 REM COURTYARD
•699 REM
•700 PRINT "WE'RE IN AN OPEN AREA
 IN THE CENTER OF"
•701 PRINT "THE CASTLE.  THE DOOR
 OF THE GREAT HALL IS JUST EAST OF
  US."C$
•702 PRINT "THE KITCHEN IS TO THE
 SOUTH AND THERE   ARE STABLES TO
 THE WEST."C$
•703 PRINT "TO THE NORTH IS THE GA
 TEHOUSE, A THICK"
•704 PRINT "BUILDING THAT SURROUND
 S AND PROTECTS THEMAIN GATE."C$
•705 PRINT "THE DIRT IN THE COURTY
 ARD IS SMOOTH, AS"
•706 PRINT "IF NO ONE HAD EVER WAL
 KED HERE. THE ONLYFOOTPRINTS ARE
 YOURS."
•715 RETURN
•717 REM
•718 REM GATEHOUSE
•719 REM
•720 PRINT "WE'RE HIGH UP IN THE G
 ATEHOUSE, BY THE"
•721 PRINT "WINDLASS THAT CONTROLS
  THE GATE.  IT'S"
•722 PRINT "STUCK--ALL YOUR STRENG
 TH CAN'T BUDGE IT.SORRY I CAN'T H
 ELP."C$
•723 PRINT "THERE ARE SHORT STAIRW
 AYS LEADING UP TO"
•724 PRINT "THE WALLS, AND ANOTHER
  STAIRWAY LEADING"
•725 PRINT "DOWN TO THE COURTYARD
 BELOW US."C$
•726 PRINT "HANGING FROM A SPIKE O
 N THE WALL IS A"
•727 PRINT "SINGLE ELEPHANT'S TUSK
 .  IT'S TOO HIGH  TO REACH."
•735 RETURN
•737 REM
•738 REM ON THE WALLS
•739 REM
•740 PRINT "WE'RE ON THE CASTLE WA
 LLS, BEHIND THE"
•741 PRINT "CRENELLATION.  WE CAN
 WALK ALL THE WAY"
•742 PRINT "AROUND THE CASTLE.  TH
 E ONLY PLACES WE"
•743 PRINT "CAN GET OFF ARE THE GA
 TEHOUSE AND THE"
•744 PRINT "TOWER THAT RISES ABOVE
  THE GREAT HALL."C$
•745 PRINT "PLEASE DON'T STAY UP H
 ERE TOO LONG."
•746 PRINT "THERE ARE BIRDS COMING
  CLOSER.  THEY"
•747 PRINT "LOOK TO ME LIKE THE IN
 SECT-EATING KIND,"
•748 PRINT "AND A FIREFLY LIKE ME
 CAN'T HIDE VERY   EASILY."
•755 RETURN
•757 REM
•758 REM GREAT HALL
•759 REM
•760 PRINT "WE'RE IN THE GREAT HAL
 L OF THE CASTLE."
•761 PRINT "A LARGE DOORWAY IN THE
  NORTH LEADS TO"
•762 PRINT "THE COURTYARD, WHILE A
```

```
 SMALL DOOR TO  THE WEST GOES TO
 THE KITCHEN."
•763 PRINT C$"THERE ARE TABLES AND
 BENCHES AROUND THE"
•764 PRINT "ROOM, A HUGE FIRE PIT
 IN THE MIDDLE, ANDA THRONE AT ONE
 END."
•765 PRINT C$"A LARGE TAPESTRY ON
 THE WALL PROCLAIMS"
•766 PRINT "THAT THIS IS THE RESID
 ENCE OF THE COUNT OF OBSCURITY."C
 $
•767 PRINT "A SMALL STAIRWAY GOES
 UP ALONG ONE WALL"
•768 PRINT "AND BEHIND THE THRONE
 THERE'S A TRAP"
•769 PRINT "DOOR THAT SOMEONE HAS
 CARELESSLY LEFT"
•770 PRINT "HALF OPEN.";
•775 RETURN
•777 REM
•778 REM COUNT'S CHAMBER
•779 REM
•780 PRINT "THE LARGE CURTAINED BE
 D LOOKS RECENTLY"
•781 PRINT "SLEPT IN, AND THE HEAR
 TH COALS ARE STILLGLOWING."C$
•782 PRINT "SMALL DOORS LEAD ONTO
 THE WALLS TO THE"
•783 PRINT "NORTH AND WEST, WHILE
 SMALL CUPBOARD"
•784 PRINT "DOORWAYS SEEM TO LURK
 BEHIND EVERY"C$"TAPESTRY."C$
•785 PRINT "A STAIRWAY IN ONE CORN
 ER GOES DOWN, AND"
•786 PRINT "A NARROW LADDER LEADS
 UP THROUGH THE"
 787 PRINT "CEILING.  I THINK I CA
 N HEAR FOOTSTEPS"
•788 PRINT "ABOVE US.  PLEASE DON'
 T GO UP THERE."
•795 RETURN
•797 REM
•798 REM TOWER LOOKOUT
•799 REM
•800 PRINT "WE'RE AT THE HIGHEST P
 OINT OF THE"C$"CASTLE."C$
•802 PRINT "IGNORE THE SOUND OF SO
 MEONE WALKING"
•803 PRINT "AROUND.  IT ISN'T ANYB
 ODY--JUST AN"
•804 PRINT "EMPTY SUIT OF ARMOR TH
 AT KEEPS WALKING"
•805 PRINT "AROUND AND AROUND THE
 EDGE OF THE TOWER."
•806 PRINT "AT LEAST I THINK IT'S
 EMPTY."C$
•807 PRINT "WOULD YOU MIND TERRIBL
 Y IF WE JUST WENT BACK DOWN THE L
 ADDER?"
•815 RETURN
•817 REM
•818 REM THE STABLES
•819 REM
•820 PRINT "THE FLOOR OF THE STABL
 E IS WOODEN, AND"
•821 PRINT "SOME OF THE BOARDS ARE
 MISSING--TEST"
•822 PRINT "EACH STEP BEFORE YOU P
 UT YOUR WEIGHT ON"
•823 PRINT "IT.  IT LOOKS LIKE A L
 ONG WAY DOWN IF"
•824 PRINT "YOU FALL THROUGH.  OR
 IS THERE SOMETHINGUNDER THE STABL
 E?":PRINT
•825 PRINT "THERE'S THAT WHINNYING
 SOUND.  IT SEEMS"
•826 PRINT "TO BE COMING FROM THAT'
 STALL, BUT THE"
•827 PRINT "STALL IS EMPTY.":PRINT
•828 PRINT "I DON'T MEAN TO SOUND
 TIMID, BUT THE"
•829 PRINT "ONLY WAY OUT OF HERE I
 S THE EAST DOOR TOTHE COURTYARD."
•835 RETURN
•837 REM
•838 REM THE DUNGEONS
•839 REM
•840 PRINT "WE'RE DOWN IN THE DUNG
 EONS, AND THE"
•841 PRINT "ONLY LIGHT DOWN HERE I
 S ME.":PRINT
•842 PRINT "AND I'M NOT ALL THAT B
 RIGHT.":PRINT
•843 PRINT "LOTS OF LITTLE CELLS E
 VERYWHERE. MOSTLY"
•844 PRINT "EMPTY.  NONE OF THEM W
 ITH ANY LIVING"
•845 PRINT "RESIDENTS.  UNLESS YOU
 COUNT RATS.":PRINT
•846 PRINT "I DON'T SEE ANY DEATH
 TRAPS, BUT THEN, ICAN'T SEE MUCH
 OF ANYTHING."
•855 RETURN
•857 REM
•858 REM THE TREASURE ROOM
```

IMPORTANT
Before typing in an *Ahoy!* program, refer to the first two pages of the program listings section.

```
•859 REM
•860 PRINT "WE MADE IT!  THERE ARE
  COINS AND INGOTS AND JEWELS EVER
 YWHERE!"
•861 PRINT:PRINT "PAINTINGS, TOO.
  AND SCULPTURES.":PRINT
•862 PRINT "NO, DON'T TAKE ANY OF
 IT!  IT'S ALL"
•863 PRINT "CURSED!  JUST THE ELEP
 HANT OF CIPANGU--"
•864 PRINT "THAT'S ALL YOU CAN TAK
 E WITH YOU, IF"
•865 PRINT "YOU WANT TO LIVE LONG
 AND PROSPER."
•875 RETURN
•900 RD(7,1)=1:RD(7,8)=1:RD(1,2)=7
 :RD(1,7)=7
•901 DS$="USING THE GOLDEN KEY, YO
 U UNLOCK THE    TURNSTILE AND OPE
 N THE GATE"
•902 RETURN
•905 IF TL(4)=0 AND TL(5)=0 AND TL
 (8)=0 THEN 910
•906 PR=XR:XR=XX:DS$=RN$(56):RETUR
 N
•910 PRINT "{SC}"DL$"DON'T BE SO C
 LUMSY!  YOU SPILLED SOME"
•911 PRINT "ALE FROM THE FLAGON ON
  THE PILLOW AND"
•912 PRINT "THE MOUSE SUDDENLY WOK
 E UP AND DRANK IT"
•913 PRINT "AND--WAIT A MINUTE--TH
 E MOUSE IS"
•914 PRINT "SPEAKING.  IT'S READIN
 G SOME ANCIENT"
•915 PRINT "WORDS WRITTEN IN THE E
 MBROIDERY OF THE"
•916 PRINT "PILLOW--AND THERE'S A
 SECRET DOOR,"
•917 PRINT "BEGINNING TO GLOW BRIG
 HTLY--IT'S THE"
•918 PRINT "TREASURE ROOM!  LET'S
 GO IN!"C$C$"(PRESS A KEY)"
•919 GOSUB 990:DS$="":RETURN
•920 PRINT "{SC}"DL$"THERE'S A CRA
 CK IN ONE FOUNDATION STONE"
•921 PRINT "AND IF YOU JUST PRY A
 LITTLE WITH THE"
•922 PRINT "BUTCHER KNIFE--YES, TH
 AT'S IT.  GET YOURBACK INTO IT."C
 $
•923 PRINT "YOU DID IT!  WE CAN CR
 AWL UNDER HERE ANDGET INSIDE.":PR
 INT
•924 PRINT "(PRESS A KEY)":DS$="":
 GOSUB 990:RETURN
•925 PR=5:XR=XX
•926 DS$="YOU CAN'T GET THROUGH WI
 TH THE ELEPHANT.I THOUGHT YOU KNE
 W THAT."
•927 DS$=DS$+C$+"ONLY THE MAIN GAT
 E WILL LET US PASS."
•928 RETURN
•937 REM
•938 REM VICTORY HANDLER
•939 REM
•940 PRINT "{SC}WE MADE IT!":PRINT
 :PRINT "WE'RE OUTSIDE THE CASTLE
 WITH THE"
•941 PRINT "EMERALD ELEPHANT OF CI
 PANGU.  NOW YOU"
•942 PRINT "CAN SEE AGAIN, AND I H
 AVE RECOVERED MY"
•943 PRINT "TRUE FORM.  I KNOW I'M
  INCREDIBLY GOOD-"
•944 PRINT "LOOKING, BUT DON'T BOT
 HER ASKING ME FOR"
•945 PRINT "A DATE--YOU'RE FINE FO
 R EXPLORING"
•946 PRINT "CASTLES, BUT I'M ENGAG
 ED TO MARRY"
•947 PRINT "SOMEBODY WITH ROYAL BL
 OOD AND A FORTUNE"
•948 PRINT "BIGGER THAN THE NATION
 AL DEBT.  YOU'VE"
•949 PRINT "BEEN SWEET, THOUGH.  T
 HANKS A BUNCH!":PRINT:PRINT "BYE!
 "
•950 POKE 198,0:END
•967 REM
•968 REM RANDOMLY CHOOSE LOCATION
 OF TREASURE ROOM
•969 REM
•970 R=INT(10*RND(9)):IF R>2 THEN
 R=3
•971 IF R=1 THEN R=0
•972 ON R GOTO 975,980,985
•975 X=2+INT(2*RND(9)):RD(10,X)=14
 :RETURN
•980 RD(12,6)=14:RETURN
•985 X=1+INT(8*RND(9)):R=13:IF RD(
 R,X)=55 THEN RD(R,X)=14:RETURN
•986 GOTO 985
•990 A=PEEK(197):IF A=64 THEN 990
•995 CM=KS(A):RETURN
•997 REM
```

```
•998 REM SET UP COMMAND WORD ARRAY
•999 REM
•1000 FOR I=0 TO 16:READ A$:CW$(I)
 =A$:NEXT
•1010 DATA "WHAT?","NORTH","SOUTH"
 ,"EAST","WEST","UP","DOWN","IN","
 OUT","BACK"
•1015 DATA "TAKE","LEAVE","PEER","
 GOT?","QUIT","HELP","HELP"
•1047 REM
•1048 REM SET UP KEYSTROKE ARRAY
•1049 REM
•1050 FOR I=0 TO 64:KS(I)=0:NEXT
•1055 FOR I=1 TO 16:READ A:KS(A)=I
 :NEXT
•1060 DATA 39,13,14,9,30,18,33,38,
 28,22,42,41,26,62,29,55
•1097 REM
•1098 REM SET UP RD (ROOM DIRECTIO
 N TABLE) AND RN$ (ROOM NAME TABLE
 )
•1099 REM
•1100 FOR I=1 TO 14:READ A$:RN$(I)
 =A$:FOR J=1 TO 8:READ A:RD(I,J)=A
 :NEXT:NEXT
•1110 DATA "CASTLE MAIN GATE",52,5
 3,3,2,51,50,53,50
•1115 DATA "MEADOW WEST OF CASTLE"
 ,1,4,51,52,51,50,51,50
•1120 DATA "LEDGE EAST OF CASTLE",
 1,54,54,51,51,51,51,50
•1125 DATA "GROVE SOUTH OF CASTLE"
 ,51,54,54,2,51,56,53,50
•1130 DATA "KITCHEN",6,4,9,6,50,13
 ,9,6
•1135 DATA "COURTYARD",7,5,9,12,7,
 5,9,7
•1140 DATA "GATEHOUSE",56,6,8,8,8,
 6,6,56
•1145 DATA "ON THE WALLS",7,10,10,
 7,57,51,7,50
•1150 DATA "GREAT HALL",6,50,50,5,
 10,13,5,6
•1155 DATA "COUNT'S CHAMBER",8,55,
 55,8,11,9,50,8
•1160 DATA "TOWER TOP",54,54,54,54
 ,57,10,50,50
•1165 DATA "THE STABLES",50,50,6,5
 0,50,55,50,6
•1170 DATA "THE DUNGEONS",55,55,55
 ,55,9,55,50,50
•1175 DATA "TREASURE ROOM",50,50,5
 0,50,50,50,50,50
•1197 REM
•1198 REM SET UP RN$ VALUES FOR IL
 LEGAL MOVEMENT DIRECTIONS
•1199 REM
•1200 FOR I=50 TO 57:READ A$:RN$(I
 )=A$:NEXT
•1210 DATA "SORRY, BUT WE CAN'T GO
  THAT WAY"
•1215 DATA "IT'S TOO STEEP FOR US
 TO CLIMB"
•1220 DATA "WE'LL JUST GET LOST IF
  WE WANDER IN THE WOODS"
•1225 DATA "I'M KNOCKING, BUT NOBO
 DY ANSWERS"
•1230 DATA "ARE YOU TRYING TO GET
•US KILLED?"
•1235 DATA "I SEARCHED THERE, FOUN
 D NOTHING, AND    CAME BACK"
•1240 DATA "WE DON'T HAVE WHAT IT
 TAKES TO GET IN   THERE"
•1245 DATA "UP FROM HERE?  DO YOU
 THINK YOU CAN FLY?"
•1297 REM
•1298 REM SET UP THE "THING" TABLE
  BY NAMING OBJECTS AND PUTTING TH
 EM IN ROOMS
•1299 REM
•1300 FOR I=0 TO 9:READ A$,B$:TN$(
 I)=A$:TS$(I)=B$:NEXT
•1310 DATA "A GOLDEN KEY","KEY","A
  LOAF OF RYE BREAD","BREAD"
•1315 DATA "A BUTCHER KNIFE","KNIF
 E","A POCKET SUNDIAL","SUNDIAL"
•1320 DATA "A DELICATELY EMBROIDER
 ED PILLOW","PILLOW"
•1325 DATA "A LIMP, MOTIONLESS MOU
 SE","MOUSE"
•1330 DATA "A PARCHMENT CODEX","CO
 DEX","AN ORANGE RIND","RIND"
•1335 DATA "A FLAGON OF ALE","FLAG
 ON"
•1340 DATA "THE EMERALD ELEPHANT O
 F CIPANGU","ELEPHANT"
•1347 REM
•1348 REM PUT EACH THING IN ITS RO
 OM
•1349 REM
•1350 FOR I=0 TO 9:READ A:TL(I)=A:
 NEXT
•1355 DATA 3,5,4,6,10,13,9,11,7,14
•1990 RETURN
```

IMPORTANT

Before typing in an *Ahoy!* program, refer to the first two pages of the program listings section.

BUG REPELLENT LINE CODES FOR EMERALD ELEPHANT

```
LINE # 1:NN      LINE # 100:DG    LINE # 372:JD    LINE # 602:ED
LINE # 2:ED      LINE # 105:KP    LINE # 373:DA    LINE # 603:PN
LINE # 3:GL      LINE # 107:JD    LINE # 374:JD    LINE # 604:DO
LINE # 4:FI      LINE # 108:LC    LINE # 375:DN    LINE # 605:MH
LINE # 5:JD      LINE # 109:JD    LINE # 380:IM    LINE # 615:IM
LINE # 6:GA      LINE # 110:CC    LINE # 387:JD    LINE # 617:JD
LINE # 7:IH      LINE # 117:JD    LINE # 388:KA    LINE # 618:ED
LINE # 8:IH      LINE # 118:EO    LINE # 389:JD    LINE # 619:JD
LINE # 9:JD      LINE # 119:JD    LINE # 390:AL    LINE # 620:NH
LINE # 10:MN     LINE # 120:EL    LINE # 397:JD    LINE # 621:CG
LINE # 20:FO     LINE # 127:JD    LINE # 398:CM    LINE # 622:DP
LINE # 25:DA     LINE # 128:CB    LINE # 399:JD    LINE # 623:BN
LINE # 27:JD     LINE # 129:JD    LINE # 400:CK    LINE # 624:EJ
LINE # 28:FH     LINE # 130:GF    LINE # 405:FF    LINE # 625:FL
LINE # 29:JD     LINE # 140:FH    LINE # 410:PG    LINE # 626:BL
LINE # 30:DO     LINE # 187:JD    LINE # 415:CA    LINE # 635:IM
LINE # 31:EJ     LINE # 188:FO    LINE # 420:CH    LINE # 637:JD
LINE # 32:MN     LINE # 189:JD    LINE # 447:JD    LINE # 638:CB
LINE # 33:CM     LINE # 190:AJ    LINE # 448:GN    LINE # 639:JD
LINE # 34:MI     LINE # 195:CA    LINE # 449:JD    LINE # 640:PD
LINE # 35:KA     LINE # 297:JD    LINE # 450:DL    LINE # 641:JB
LINE # 60:DN     LINE # 298:AF    LINE # 455:OJ    LINE # 642:LB
LINE # 61:FI     LINE # 299:JD    LINE # 460:DB    LINE # 643:GM
LINE # 62:OC     LINE # 300:EC    LINE # 467:JD    LINE # 644:IH
LINE # 63:CH     LINE # 302:JD    LINE # 468:JJ    LINE # 645:DM
LINE # 64:KB     LINE # 303:GE    LINE # 469:JD    LINE # 646:EP
LINE # 65:AF     LINE # 304:JD    LINE # 470:HJ    LINE # 647:CE
LINE # 66:BL     LINE # 310:HN    LINE # 475:KG    LINE # 655:IM
LINE # 67:KA     LINE # 315:FE    LINE # 480:MF    LINE # 657:JD
LINE # 68:BE     LINE # 320:MA    LINE # 485:CA    LINE # 658:JP
LINE # 69:DM     LINE # 325:LE    LINE # 497:JD    LINE # 659:JD
LINE # 70:FA     LINE # 342:JD    LINE # 498:CG    LINE # 660:IC
LINE # 71:GO     LINE # 343:CO    LINE # 499:JD    LINE # 661:JN
LINE # 72:PJ     LINE # 344:JD    LINE # 500:AO    LINE # 662:JH
LINE # 73:JA     LINE # 345:IN    LINE # 501:NK    LINE # 663:DA
LINE # 74:IN     LINE # 346:JD    LINE # 502:OI    LINE # 664:KN
LINE # 75:KN     LINE # 347:AL    LINE # 503:MB    LINE # 665:OM
LINE # 76:ML     LINE # 348:DB    LINE # 504:HK    LINE # 666:OJ
LINE # 77:AL     LINE # 349:JD    LINE # 505:DC    LINE # 675:IM
LINE # 78:FK     LINE # 350:IJ    LINE # 506:KO    LINE # 677:JD
LINE # 79:CM     LINE # 355:OO    LINE # 507:EL    LINE # 678:JK
LINE # 80:GH     LINE # 360:AE    LINE # 547:JD    LINE # 679:JD
LINE # 81:DG     LINE # 362:JD    LINE # 548:KA    LINE # 680:HM
LINE # 87:JD     LINE # 363:DM    LINE # 549:JD    LINE # 681:LG
LINE # 88:BN     LINE # 364:JD    LINE # 550:BB    LINE # 682:KH
LINE # 89:JD     LINE # 365:CG    LINE # 555:PI    LINE # 683:JK
LINE # 90:JP     LINE # 367:JD    LINE # 560:KI    LINE # 684:GI
LINE # 97:JD     LINE # 368:OP    LINE # 565:IM    LINE # 685:BO
LINE # 98:GA     LINE # 369:JD    LINE # 597:JD    LINE # 686:OJ
LINE # 99:JD     LINE # 370:FF    LINE # 598:PM    LINE # 687:II
                                  LINE # 599:JD    LINE # 688:NP
                                  LINE # 600:NB    LINE # 689:HH
                                  LINE # 601:OG    LINE # 690:BP
```

```
LINE # 691:IA
LINE # 692:JL
LINE # 693:GH
LINE # 695:IM
LINE # 697:JD
LINE # 698:GG
LINE # 699:JD
LINE # 700:BG
LINE # 701:AK
LINE # 702:MM
LINE # 703:FL
LINE # 704:OB
LINE # 705:MJ
LINE # 706:IO
LINE # 715:IM
LINE # 717:JD
LINE # 718:FA
LINE # 719:JD
LINE # 720:MI
LINE # 721:KK
LINE # 722:BJ
LINE # 723:AG
LINE # 724:LM
LINE # 725:NC
LINE # 726:MI
LINE # 727:CK
LINE # 735:IM
LINE # 737:JD
LINE # 738:JD
LINE # 739:JD
LINE # 740:MC
LINE # 741:OO
LINE # 742:MN
LINE # 743:OE
LINE # 744:CD
LINE # 745:FB
LINE # 746:PL
LINE # 747:DI
LINE # 748:BH
LINE # 755:IM
LINE # 757:JD
LINE # 758:DF
LINE # 759:JD
LINE # 760:OE
LINE # 761:CD
LINE # 762:EO
LINE # 763:GN
LINE # 764:BH
LINE # 765:GJ
LINE # 766:JH
LINE # 767:GM
LINE # 768:KG
LINE # 769:PM
LINE # 770:MA
LINE # 775:IM
LINE # 777:JD
LINE # 778:LB
LINE # 779:JD
LINE # 780:PA
LINE # 781:KB
LINE # 782:AE
LINE # 783:MK
LINE # 784:DB
LINE # 785:IC
LINE # 786:OI
LINE # 787:EI
LINE # 788:FL
LINE # 795:IM
LINE # 797:JD
LINE # 798:EP
LINE # 799:JD
LINE # 800:HF
LINE # 802:GE
LINE # 803:GI
LINE # 804:MH
LINE # 805:PB
LINE # 806:FG
LINE # 807:PI
LINE # 815:IM
LINE # 817:JD
LINE # 818:GD
LINE # 819:JD
LINE # 820:HJ
LINE # 821:FI
LINE # 822:NK
LINE # 823:BD
LINE # 824:BK
LINE # 825:CC
LINE # 826:OB
LINE # 827:PE
LINE # 828:KO
LINE # 829:DA
LINE # 835:IM
LINE # 837:JD
LINE # 838:MK
LINE # 839:JD
LINE # 840:HC
LINE # 841:PI
LINE # 842:MG
LINE # 843:FO
LINE # 844:GI
LINE # 845:IL
LINE # 846:NE
LINE # 855:IM
LINE # 857:JD
LINE # 858:CN
LINE # 859:JD
LINE # 860:IO
LINE # 861:FG
LINE # 862:DD
LINE # 863:PJ
LINE # 864:EH
LINE # 865:MJ
LINE # 875:IM
LINE # 900:CN
LINE # 901:AL
LINE # 902:IM
LINE # 905:BP
LINE # 906:MH
LINE # 910:ML
LINE # 911:AH
LINE # 912:BD
LINE # 913:HH
LINE # 914:EA
LINE # 915:NL
LINE # 916:NO
LINE # 917:HA
LINE # 918:PA
LINE # 919:AK
LINE # 920:JH
LINE # 921:LC
LINE # 922:MM
LINE # 923:EO
LINE # 924:BB
LINE # 925:LP
LINE # 926:ID
LINE # 927:CJ
LINE # 928:IM
LINE # 937:JD
LINE # 938:PA
LINE # 939:JD
LINE # 940:BE
LINE # 941:KO
LINE # 942:BC
LINE # 943:OO
LINE # 944:ML
LINE # 945:NF
LINE # 946:AH
LINE # 947:PD
LINE # 948:MB
LINE # 949:CC
LINE # 950:GG
LINE # 967:JD
LINE # 968:LG
LINE # 969:JD
LINE # 970:GA
LINE # 971:KC
LINE # 972:JG
LINE # 975:HF
LINE # 980:IB
LINE # 985:KE
LINE # 986:DI
LINE # 990:EM
LINE # 995:EG
LINE # 997:JD
LINE # 998:PI
LINE # 999:JD
LINE # 1000:EL
LINE # 1010:FK
LINE # 1015:IB
LINE # 1047:JD
LINE # 1048:HD
LINE # 1049:JD
LINE # 1050:NI
LINE # 1055:AI
LINE # 1060:PJ
LINE # 1097:JD
LINE # 1098:NH
LINE # 1099:JD
LINE # 1100:JI
LINE # 1110:PP
LINE # 1115:PH
LINE # 1120:IP
LINE # 1125:FO
LINE # 1130:ML
LINE # 1135:GM
LINE # 1140:MK
LINE # 1145:CJ
LINE # 1150:IF
LINE # 1155:JP
LINE # 1160:BK
LINE # 1165:DF
LINE # 1170:AD
LINE # 1175:AD
LINE # 1197:JD
LINE # 1198:LF
LINE # 1199:JD
LINE # 1200:DH
LINE # 1210:IM
LINE # 1215:PK
LINE # 1220:JI
LINE # 1225:KG
LINE # 1230:KF
LINE # 1235:HB
LINE # 1240:AC
LINE # 1245:GM
LINE # 1297:JD
LINE # 1298:DJ
LINE # 1299:JD
LINE # 1300:EA
LINE # 1310:PI
LINE # 1315:IJ
```

IMPORTANT
Before typing in an *Ahoy!* program, refer to the first two pages of the program listings section.

```
LINE # 1320:LO    LINE # 1348:CF
LINE # 1325:FD    LINE # 1349:JD
LINE # 1330:OA    LINE # 1350:CF
LINE # 1335:AF    LINE # 1355:BE
LINE # 1340:BA    LINE # 1990:IM
LINE # 1347:JD    LINES: 429
```

BAM READ AND PRINT

FROM PAGE 96

```
•1 GOTO10
•2 BAM DISPLAY & PRINT
•3 FOR COMMODORE 64,
•4 -1541 DISK DRIVE &
•5 -1525/1526 PRINTER
•6 BY M A KEVELSON PE
•7 APRIL 30, 1984
•10 CLR
•15 POKE 53280,8:POKE 53281,1:PRIN
T"{BK}";
•20 SP$="          ":NU$="1234567890
"
•25 PRINT "{SC}        BAM DISPLAY BY
 MORTON KEVELSON"
•30 PRINT"{CD}{CD}{CD}{CD}{CD}
      {RV}S{RO}CREEN OR {RV}P{RO}
RINTER"
•40 GETPR$:IFPR$=""GOTO40
•50 DIM SE%(35,21)
•60 OPEN15,8,15:PRINT#15,"I0"
•70 OPEN1,8,2,"#"
•80 PRINT#15,"U1:";2;0;18;0
•90 PRINT#15,"B-P:"2;4
•100 FOR TR=1 TO 35:PRINT"{SC}TRAC
K";TR
•110 GET#1,CO$:SE%(TR,21)=ASC(RIGH
T$(CHR$(0)+CO$,1))
•120 FOR GR=0 TO 2:GET#1,GR$
•130 G%=ASC(RIGHT$(CHR$(0)+GR$,1))
•140 FOR BT=0 TO 7
•150 CK=2^BT AND G%
•160 BL=BT+8*GR
•170 IF BL>20 GOTO200
•180 IF CK THEN SE%(TR,BL)=1:GOTO2
00
•190 SE%(TR,BL)=0
•200 NEXT BT,GR,TR
•210 CLOSE1:CLOSE15
•220 IFPR$="P"GOTO400
•230 PRINT"    <<<<<<<<<< TRACK  N
UMBER >>>>>>>>>>"
•240 PRINT"    ";:FORI=1TO3:PRINTS
P$;STR$(I);:NEXTI:PRINT
•250 PRINT"    ";:FORI=1TO3:PRINTN
U$;:NEXTI:PRINT"12345"
•260 PRINT"BLK#"
•270 FOR BL=0 TO 19
•280 T=34+18*(BL>18)+11*(BL=18)+5*
(BL=17)
•290 PRINTRIGHT$("   "+STR$(BL)+"
",4);
•300 FOR TR=1 TO T
•310 PRINTMID$(STR$(SE%(TR,BL)),2)
;
•320 NEXT TR
•330 PRINTMID$(STR$(SE%(TR,BL)),2)
•340 NEXT BL
•350 PRINT" 20 ";
•360 FOR TR=1 TO 17
•370 PRINTMID$(STR$(SE%(TR,20)),2)
;
•380 NEXT TR
•390 GOTO540
•400 OPEN4,4:PRINT#4,
•410 PRINT#4,"    <<<<<<<<<< TRACK
  NUMBER >>>>>>>>>>"
•420 PRINT#4,"    ";:FORI=1TO3:PRI
NT#4,SP$;STR$(I);:NEXTI:PRINT#4,
•430 PRINT#4,"    ";:FORI=1TO3:PRI
NT#4,NU$;:NEXTI:PRINT#4,"12345"
•440 PRINT#4,"BLK#"
•450 FOR BL=0 TO 20
•460 T=34+18*(BL>18)+11*(BL=18)+5*
(BL=17)
•470 PRINT#4,RIGHT$("   "+STR$(BL)
+" ",4);
•480 FOR TR=1 TO T
•490 PRINT#4,MID$(STR$(SE%(TR,BL))
,2);
•500 NEXT TR
•510 PRINT#4,MID$(STR$(SE%(TR,BL))
,2)
•520 NEXT BL
•530 PRINT#4,:PRINT#4,:CLOSE4:GOTO
•230
•540 GETA$:IFA$=""GOTO540
•550 PRINT:PRINT"HIT A KEY WHEN NE
XT DISK IS READY"
•560 GETA$:IFA$=""GOTO560
•570 GOTO10
```

C-64 BUG REPELLENT LINE CODES FOR BAM READ AND PRINT

```
LINE # 1:PH       LINE # 2:NE
```

```
LINE # 3:PE
LINE # 4:DE
LINE # 5:EB
LINE # 6:BP
LINE # 7:PA
LINE # 10:JO
LINE # 15:DJ
LINE # 20:NE
LINE # 25:MF
LINE # 30:KF
LINE # 40:MG
LINE # 50:JO
LINE # 60:FA
LINE # 70:CL
LINE # 80:PP
LINE # 90:PI
LINE # 100:CM
LINE # 110:CJ
LINE # 120:AL
LINE # 130:DE
LINE # 140:NF
LINE # 150:NJ
LINE # 160:AN
LINE # 170:FF
LINE # 180:MM
LINE # 190:FL
LINE # 200:MP
LINE # 210:BB
LINE # 220:NJ
LINE # 230:MN
LINE # 240:PP
LINE # 250:BE
LINE # 260:OG
LINE # 270:AE
LINE # 280:AE
LINE # 290:FH
LINE # 300:AJ
LINE # 310:AN
LINE # 320:CP
LINE # 330:ME
LINE # 340:BD
LINE # 350:LN
LINE # 360:BN
LINE # 370:KB
LINE # 380:CP
LINE # 390:CN
LINE # 400:FM
LINE # 410:LJ
LINE # 420:JJ
LINE # 430:PM
LINE # 440:FI
LINE # 450:PK
LINE # 460:AE
LINE # 470:HP
LINE # 480:AJ
LINE # 490:BN
LINE # 500:CP
LINE # 510:PG
LINE # 520:BD
LINE # 530:GN
LINE # 540:FK
LINE # 550:EH
LINE # 560:FE
LINE # 570:PH
LINES: 66
```

VIC 40 OPERATING SYSTEM

FROM PAGE 90 ENTRY PROGRAM

```
•2 POKE36879,27:POKE56,33
•10 PRINT"{SC}{CD}{BL} VIC 40 ENTR
Y PROGRAM"
•20 PRINT"{CD}   PLEASE ENTER THE"
•30 PRINT"{CD}  DATA {RV}EXACTLY{R
O} AS IT"
•40 PRINT"{CD}        APPEARS{CD}"
•50 CLR:Z=4:Y=3:W=2:X=1:A=4096:B=2
56:C=16:D=48:E=7:F=10:G=6:H=100:K
=.
•52 CLOSEX:OPENX,.:PRINT".:";:INPU
T#X,A$:PRINT:IFLEFT$(A$,1)="F"THE
N100
•53 L$=LEFT$(A$,Z):FORI=XTOZ
•60 Q=ASC(MID$(A$,I,1))-D:L(I)=Q+E
*(Q>F):NEXT:L=L(X)*A+L(W)*B+L(Y)*
C+L(Z)
•70 FORI=.TOE:B$=MID$(A$,G+I*Y,W):
R=ASC(LEFT$(B$,1))-D:S=ASC(RIGHT$
(B$,1))-D
•80 R=R+E*(R>F):S=S+E*(S>F):P=R*C+
S:POKEL+I,P:K=K+P:NEXT
•90 CH$=STR$(K-INT(K/H)*H):IFVAL(C
H$)<10THENCH$=" 0"+RIGHT$(CH$,1)
•95 PRINT"{CD}CHECKSUM IS"CH$"{CD}
":GOTO50
•100 PRINT"{SC}{CD}{CD}WANT VIC 40
 SAVED TO"
•110 PRINT"{CD}DISK OR TAPE (D/T)
? ";
•120 GETR$:IFR$=""THEN120
•130 IFR$="T"THEND=1:GOTO160
•140 IFR$="D"THEND=8:GOTO160
•150 GOTO100
•160 PRINTR$:PRINT"{CD}{CD}HIT A K
EY WHEN READY"
•170 PRINT"{CD}FOR SAVE TO BEGIN."
•180 GETR$:IFR$=""THEN180
•190 PRINT"{SC}{CD}{CD}{CD}POKE43,
136:POKE44,34"
•200 PRINT"{HM}{CD}{CD}{CD}{CD}{CD
}{CD}POKE45,105:POKE46,46"
•210 PRINT"{HM}{CD}{CD}{CD}{CD}{CD
}{CD}{CD}{CD}{CD}SAVE"CHR$(34)"VI
C 40"CHR$(34)","D"{HM}"
•220 POKE198,3:POKE631,13:POKE632,
13:POKE633,13
```

BUG REPELLENT LINE CODES FOR VIC 40—ENTRY PROGRAM

```
LINE # 2:AO
LINE # 10:MD
LINE # 20:OD
LINE # 30:LM
LINE # 40:OM
LINE # 50:EM
LINE # 52:LC
LINE # 53:GE
LINE # 60:MC
LINE # 70:FH
LINE # 80:JH
LINE # 90:EM
LINE # 95:GI
LINE # 100:FI
LINE # 110:MN
LINE # 120:HA
LINE # 130:KK
LINE # 140:GN
LINE # 150:CF
LINE # 160:OF
LINE # 170:FG
LINE # 180:IO
LINE # 190:KJ
LINE # 200:CM
LINE # 210:GC
LINE # 220:CC
LINES: 26
```

MAIN PROGRAM

IMPORTANT

Before typing in an *Ahoy!* program, refer to the first two pages of the program listings section.

Note: numbers on white background are checksum values. *Do not enter!* See article for explanation.

```
.:2288 20 20 2C 2A A9 93 20 D2 08
.:2290 FF A9 1F 20 35 2D A9 0F 69
.:2298 8D 0E 90 20 5F 2D EA EA 39
.:22A0 EA EA A0 00 B9 87 2C 20 24
.:22A8 D2 FF C8 C0 65 D0 F5 60 07
.:22B0 AD 03 2D CD 02 2D F0 0C 25
.:22B8 8D 02 2D C9 0F 90 02 A9 19
.:22C0 0F 8D 0E 90 AD 06 2D CD 43
.:22C8 01 2D F0 06 8D 01 2D 20 11
.:22D0 EF 2C AD 05 2D CD 04 2D 60
.:22D8 F0 0D 8D 04 2D A0 8E B0 21
.:22E0 02 A0 0E 98 20 D2 FF A0 85
.:22E8 0C AD FF 02 F0 02 A0 0E 58
.:22F0 8C 04 2D 8C 05 2D 20 EA 45
.:22F8 FF 4C FB 2C 60 93 22 3B 62
.:2300 22 55 55 44 44 33 00 02 93
.:2308 05 35 57 55 35 00 46 45 22
.:2310 66 55 55 66 00 02 05 34 33
.:2318 44 45 32 00 16 15 35 55 68
.:2320 55 36 00 07 04 77 54 64 53
.:2328 77 00 17 24 24 76 24 24 04
.:2330 00 02 05 34 55 35 13 60 12
.:2338 45 45 67 55 55 55 00 07 03
.:2340 22 02 22 22 27 00 01 11 61
.:2348 01 11 15 52 20 45 45 56 77
.:2350 66 55 55 00 64 24 24 24 80
.:2358 24 77 00 05 07 57 75 55 56
.:2360 55 00 07 05 65 55 55 55 53
.:2368 00 02 05 25 55 55 22 00 48
.:2370 06 05 65 56 64 44 40 02 32
.:2378 05 35 55 37 13 10 06 05 44
.:2380 55 66 45 45 00 03 04 32 82
.:2388 61 11 66 00 27 22 72 22 37
.:2390 22 32 00 05 05 55 55 55 49
.:2398 37 00 05 05 55 55 77 22 88
.:23A0 00 05 05 55 57 77 55 00 86
.:23A8 05 07 52 27 25 55 00 05 60
.:23B0 05 57 52 32 12 60 07 01 46
.:23B8 73 36 44 77 00 66 44 44 94
.:23C0 44 44 66 00 33 22 22 77 76
.:23C8 66 77 00 66 22 22 22 22 59
.:23D0 66 00 00 22 77 55 55 00 25
.:23D8 00 00 00 33 44 33 00 00 70
.:23E0 00 00 00 00 00 00 00 22 34
.:23E8 22 22 22 00 22 00 55 55 06
.:23F0 00 00 00 00 00 00 55 77 04
.:23F8 55 77 55 00 22 77 66 33 95
.:2400 77 22 00 55 11 22 22 44 91
.:2408 55 00 66 44 66 22 55 33 27
.:2410 00 11 22 00 00 00 00 00 51
.:2418 11 22 22 22 22 11 00 44 38
.:2420 22 22 22 22 44 00 00 55 89
.:2428 22 77 22 55 00 00 00 22 06
.:2430 77 22 00 00 00 00 00 00 53
.:2438 11 11 22 00 00 00 77 00 87
.:2440 00 00 00 00 00 00 22 22 68
.:2448 00 11 11 33 66 44 44 00 23
.:2450 77 55 55 55 55 77 00 11 95
.:2458 33 55 11 11 11 00 22 55 06
.:2460 11 22 44 77 00 77 11 22 08
.:2468 11 55 22 00 55 55 77 11 42
.:2470 11 11 00 77 44 66 11 55 25
.:2478 22 00 33 44 66 55 55 22 59
.:2480 00 77 11 11 22 22 22 00 55
.:2488 77 55 22 55 55 77 00 22 61
.:2490 55 55 33 11 66 00 00 00 40
.:2498 22 00 22 00 00 00 00 22 02
.:24A0 00 22 22 44 00 33 44 88 91
.:24A8 44 33 00 00 00 77 00 77 57
.:24B0 00 00 00 CC 22 11 22 CC 93
.:24B8 00 22 55 11 22 00 22 00 04
.:24C0 00 00 00 FF 00 00 00 20 87
.:24C8 52 57 77 52 57 00 64 54 41
.:24D0 64 54 54 64 04 20 50 40 48
.:24D8 4F 50 20 00 60 50 5F 50 42
.:24E0 50 60 00 70 4F 70 40 40 07
.:24E8 70 00 70 40 40 60 4F 40 91
.:24F0 00 24 54 44 54 54 34 04 12
.:24F8 52 52 72 52 52 52 02 70 38
.:2500 20 20 28 24 74 04 14 14 00
.:2508 12 11 50 20 00 54 54 64 15
.:2510 68 50 50 00 48 48 48 48 52
.:2518 48 78 0F 58 78 74 56 52 99
.:2520 51 01 71 51 52 56 54 58 16
.:2528 08 2F 58 58 58 58 28 08 55
.:2530 6F 51 51 61 41 41 01 20 33
.:2538 50 56 5F 7F 36 00 60 50 18
.:2540 50 60 50 5F 00 30 45 27 07
.:2548 17 12 62 00 78 28 28 28 79
.:2550 28 28 08 50 50 50 51 52 91
.:2558 74 04 59 59 56 56 76 29 29
.:2560 09 50 50 56 79 79 56 00 83
.:2568 50 72 22 77 57 52 00 51 97
.:2570 51 71 21 21 21 01 70 10 22
.:2578 36 6F 46 70 00 44 44 44 51
.:2580 FF 44 44 44 44 88 44 88 67
.:2588 44 88 44 44 44 44 44 44 12
.:2590 44 44 00 00 11 FF 55 55 78
.:2598 00 DF 6F B7 D7 63 B3 D1 19
.:25A0 00 00 00 00 00 00 00 CC 04
.:25A8 CC CC CC CC CC CC 00 00 24
.:25B0 00 FF FF FF FF FF 00 00 75
.:25B8 00 00 00 00 00 00 00 00 00
```

```
.:25C0 00 00 FF 88 88 88 88 88 35
.:25C8 88 88 AA 55 AA 55 AA 55 37
.:25D0 AA 11 11 11 11 11 11 11 89
.:25D8 00 00 00 AA 55 AA 55 BF 01
.:25E0 6F DE BE 6C DC B8 11 11 69
.:25E8 11 11 11 11 11 44 44 44 89
.:25F0 77 44 44 44 00 00 00 33 74
.:25F8 33 33 33 44 44 44 77 00 76
.:2600 00 00 00 00 00 CC 44 44 40
.:2608 44 00 00 00 00 00 FF FF 78
.:2610 00 00 00 77 44 44 44 44 91
.:2618 44 44 FF 00 00 00 00 00 91
.:2620 00 FF 44 44 44 44 44 44 63
.:2628 CC 44 44 44 88 88 88 88 52
.:2630 88 88 88 CC CC CC CC CC 28
.:2638 CC CC 33 33 33 33 33 33 14
.:2640 33 FF FF 00 00 00 00 00 61
.:2648 FF FF FF 00 00 00 00 00 65
.:2650 00 00 00 FF FF FF 11 11 99
.:2658 31 21 A1 41 0F 00 00 00 23
.:2660 CC CC CC CC 33 33 33 33 20
.:2668 00 00 00 44 44 44 CC 00 08
.:2670 00 00 CC CC CC CC 00 00 16
.:2678 00 CC CC CC CC 33 33 33 69
.:2680 07 19 14 17 FF CC E1 2A 01
.:2688 E0 26 20 20 85 FB 86 FC 96
.:2690 A2 00 86 FD 86 FE 86 FF 26
.:2698 18 A5 FB F0 17 46 FB 90 68
.:26A0 0D A5 FC 18 65 FE 85 FE 96
.:26A8 A5 FD 65 FF 85 FF 06 FC 20
.:26B0 26 FD 90 E5 A6 FE A5 FF 04
.:26B8 60 A5 D6 A2 28 20 8C 26 87
.:26C0 A8 8A 69 A0 85 D1 98 69 70
.:26C8 1E 85 D2 A6 D6 B5 D9 F0 91
.:26D0 0B B5 DA F0 07 A0 27 84 88
.:26D8 D5 A4 D3 60 A0 4F D0 F7 78
.:26E0 48 85 D7 A5 9A C9 03 F0 83
.:26E8 03 4C 85 F2 78 8A 48 98 36
.:26F0 48 A9 00 85 D0 A4 D3 A5 22
.:26F8 D7 10 03 4C 4F 27 C9 0D 42
.:2700 D0 03 4C 02 29 C9 20 90 07
.:2708 10 C9 60 90 04 29 DF D0 33
.:2710 02 29 3F 20 B8 E6 4C 78 48
.:2718 2A A6 D8 F0 03 4C 7C 2A 09
.:2720 C9 14 D0 03 4C 18 29 A6 39
.:2728 D4 F0 03 4C 7C 2A C9 12 16
.:2730 D0 02 85 C7 C9 13 D0 03 73
.:2738 20 FB 28 C9 1D D0 03 4C 40
.:2740 EC 29 C9 11 D0 03 4C FB 33
.:2748 29 20 21 28 4C CE 2A 29 11
.:2750 7F C9 7F D0 02 A9 5E C9 29
.:2758 20 90 05 09 40 4C 78 2A 92
.:2760 C9 0D D0 08 4C 02 29 09 58
.:2768 40 4C 7C 2A C9 14 D0 03 38
.:2770 4C 97 29 A6 D4 D0 F0 A6 60
.:2778 D8 D0 EC C9 11 D0 03 4C 65
.:2780 03 2A C9 12 D0 02 86 C7 07
.:2788 C9 1D D0 03 4C 0E 2A C9 74
.:2790 13 D0 03 20 BD 28 09 80 28
.:2798 20 21 28 C9 80 D0 03 4C 21
.:27A0 23 2A 4C D7 2A 85 D7 29 99
.:27A8 7F A2 07 20 8C 26 86 03 43
.:27B0 69 23 85 04 A5 D6 A2 07 25
.:27B8 20 8C 26 86 00 85 01 A5 43
.:27C0 D3 AA 29 01 85 02 8A 4A 70
.:27C8 A2 B0 20 8C 26 A8 8A 65 55
.:27D0 00 85 00 98 65 01 85 01 21
.:27D8 A5 00 69 E0 85 00 A5 01 93
.:27E0 69 10 85 01 A0 06 A9 0F 05
.:27E8 A6 02 F0 02 A9 F0 31 00 68
.:27F0 91 00 88 10 F1 A0 06 B1 81
.:27F8 03 AE FF 02 F0 04 4A 4A 26
.:2800 4A 4A 29 0F A6 D7 10 02 03
.:2808 49 0F A6 02 D0 04 0A 0A 88
.:2810 0A 0A 11 00 91 00 88 10 34
.:2818 DE 20 B9 26 A5 D7 91 D1 11
.:2820 60 20 12 E9 30 09 8A A2 36
.:2828 F0 9D FF 93 CA D0 FA 60 55
.:2830 A9 90 85 03 A9 11 85 04 72
.:2838 A0 0D A5 DA F0 02 A0 06 64
.:2840 A9 00 91 03 88 10 FB 18 44
.:2848 A9 B0 65 03 85 03 A9 00 54
.:2850 65 04 C9 1F 90 E0 A2 E0 91
.:2858 86 03 A2 10 86 04 A0 07 20
.:2860 A2 00 A5 DA D0 02 A0 0E 29
.:2868 B1 03 81 03 E6 03 D0 02 55
.:2870 E6 04 A5 03 C9 9F D0 F0 10
.:2878 A5 04 C9 1E D0 EA E6 03 75
.:2880 A0 28 A5 DA D0 02 A0 50 33
.:2888 B1 03 81 03 E6 03 D0 02 55
.:2890 E6 04 98 18 65 03 C9 88 51
.:2898 D0 EE A5 04 C9 22 D0 E8 90
.:28A0 A9 20 91 03 88 10 FB C6 50
.:28A8 D6 A2 00 B5 DA 95 D9 E8 73
.:28B0 E0 19 D0 F7 A9 FF 95 D9 94
.:28B8 A5 D9 F0 EB 60 A0 10 84 61
.:28C0 04 A0 E0 84 03 A0 00 98 35
.:28C8 91 03 E6 03 D0 02 E6 04 25
.:28D0 A6 03 E0 A0 D0 F2 A6 04 73
.:28D8 E0 1E D0 EC A9 20 91 03 47
.:28E0 E6 03 D0 02 E6 04 A6 03 46
.:28E8 E0 88 D0 F2 A6 04 E0 22 38
.:28F0 D0 EC A9 FF A2 1A 95 D9 22
.:28F8 CA 10 FB A2 00 86 D3 86 10
.:2900 D6 60 A2 00 86 D3 86 D4 63
.:2908 86 D8 86 C7 A6 D6 E8 B5 76
```

> **IMPORTANT**
> Before typing in an *Ahoy!* program, refer to the first two pages of the program listings section.

```
.:2910 D9 F0 FB 86 D6 4C 8C 2A 14
.:2918 A4 D3 D0 15 A6 D6 B5 D9 82
.:2920 F0 03 4C 0E 2A C6 D6 A0 47
.:2928 28 84 D3 20 B9 26 10 01 55
.:2930 C8 B1 D1 88 91 D1 C8 C0 68
.:2938 27 90 F5 A6 D6 B5 DA D0 15
.:2940 0C B1 D1 88 91 D1 C8 C8 88
.:2948 C0 50 90 F5 88 A9 20 91 43
.:2950 D1 C6 D3 A5 D8 48 A5 D3 47
.:2958 48 A5 D6 48 A4 D3 B1 D1 84
.:2960 20 A5 27 20 A8 2A D0 F4 30
.:2968 B5 D9 D0 06 20 B9 26 18 91
.:2970 90 EA 68 85 D6 68 85 D3 77
.:2978 68 85 D8 A6 D6 4C 8C 2A 91
.:2980 85 FC 29 3F 06 FC 24 FC 35
.:2988 10 02 09 80 90 04 A6 D4 81
.:2990 D0 04 70 02 09 40 60 A6 61
.:2998 D6 B5 DA F0 0D A5 D3 C9 43
.:29A0 27 D0 03 4C DC E6 A0 27 75
.:29A8 D0 02 A0 4F B1 D1 C9 20 68
.:29B0 D0 F1 88 B1 D1 C8 91 D1 25
.:29B8 88 C4 D3 D0 F5 A9 20 91 42
.:29C0 D1 E6 D8 E6 D8 4C 53 29 01
.:29C8 A0 06 A9 0F A6 02 D0 02 28
.:29D0 A9 F0 51 00 91 00 88 10 87
.:29D8 F1 A5 CF 49 01 85 CF 60 23
.:29E0 A5 CC D0 07 A5 CF F0 03 99
.:29E8 20 C8 29 60 20 E0 29 E6 96
.:29F0 D3 A5 D3 C9 28 D0 09 A9 14
.:29F8 00 85 D3 20 E0 29 E6 D6 85
.:2A00 4C 7B 29 A5 D6 F0 F9 20 40
.:2A08 E0 29 C6 D6 10 F2 20 E0 91
.:2A10 29 A5 D3 D0 0A A5 D6 F0 54
.:2A18 E7 C6 D6 A9 28 85 D3 C6 94
·2A20 D3 10 DD 20 52 FD 20 18 71
.:2A28 E5 4C DC E6 20 01 2E A9 03
.:2A30 0E 8D 35 2A A9 19 99 00 97
.:2A38 10 C8 18 69 0B C9 EA 90 35
.:2A40 F5 EE 35 2A C0 DC 90 EC 70
.:2A48 A0 05 B9 80 26 99 00 90 13
.:2A50 88 10 F7 A0 04 B9 85 26 19
.:2A58 99 23 03 88 D0 F7 78 A9 71
.:2A60 B9 8D 14 03 8D 91 02 A9 06
.:2A68 2A 8D 15 03 A9 47 8D 18 12
.:2A70 03 A9 2E 85 2C 4C 30 2D 64
.:2A78 A6 C7 F0 02 09 80 20 A5 41
.:2A80 27 20 A8 2A D0 0D B5 D8 99
.:2A88 F0 02 94 D9 E0 19 90 03 03
.:2A90 20 30 28 20 B9 26 B1 D1 61
.:2A98 20 A5 27 A2 02 86 CD A6 05
.:2AA0 D8 F0 02 C6 D8 4C DC E6 98
.:2AA8 E6 D3 A5 D3 C9 28 90 08 10
.:2AB0 E6 D6 A6 D6 A0 00 84 D3 27
.:2AB8 60 4C B0 22 A5 CC D0 0B 70
.:2AC0 C6 CD D0 07 A9 14 85 CD 45
.:2AC8 20 C8 29 4C EF EA C9 0E 37
.:2AD0 D0 C1 8D FF 02 F0 BC C9 28
.:2AD8 8E D0 B8 E8 8E FF 02 F0 05
.:2AE0 B2 A5 99 D0 0A A5 D6 85 26
.:2AE8 C9 A5 D3 85 CA 10 0D C9 42
.:2AF0 03 F0 03 4C 2A F2 85 D0 47
.:2AF8 A5 D5 85 C8 98 48 8A 48 45
.:2B00 A5 D0 F0 3A A4 D3 B1 D1 32
.:2B08 20 80 29 E6 D3 20 B8 E6 88
.:2B10 C4 C8 D0 17 A9 00 85 D0 37
.:2B18 A9 0D A6 99 E0 03 F0 06 74
.:2B20 A6 9A E0 03 F0 03 20 E0 46
.:2B28 26 A9 0D 85 D7 68 AA 68 46
.:2B30 A8 A5 D7 C9 DE D0 02 A9 50
.:2B38 FF 18 60 20 E0 26 A5 C6 32
.:2B40 85 CC 8D 92 02 F0 F7 78 33
.:2B48 20 E4 29 20 CF E5 C9 83 01
.:2B50 D0 10 A2 09 78 86 C6 BD 36
.:2B58 F3 ED 9D 76 02 CA D0 F7 14
.:2B60 F0 DC C9 0D D0 D5 A6 D6 75
.:2B68 B5 D9 D0 01 CA 86 D6 20 89
.:2B70 B9 26 A4 D5 84 D0 B1 D1 26
.:2B78 C9 20 D0 03 88 D0 F7 C8 35
.:2B80 84 C8 A0 00 8C 92 02 84 12
.:2B88 D4 A5 D6 C5 C9 F0 06 A9 04
.:2B90 00 85 D3 F0 0B A5 CA 85 95
.:2B98 D3 C5 C8 90 03 4C 14 2B 94
.:2BA0 4C 04 2B 20 73 00 20 AC 74
.:2BA8 2B 4C AE C7 D0 01 60 E9 30
.:2BB0 80 B0 03 4C A5 C9 C9 23 85
.:2BB8 B0 07 C9 17 F0 06 4C F7 76
.:2BC0 C7 4C 0E C8 20 73 00 20 68
.:2BC8 EB D7 20 07 2D EA B0 45 13
.:2BD0 C9 80 90 41 29 7F 85 15 60
.:2BD8 A5 D6 48 A5 D3 48 A5 00 64
.:2BE0 48 A5 01 48 A5 02 48 A0 09
.:2BE8 FF A5 14 38 E9 28 85 14 22
.:2BF0 A5 15 E9 00 85 15 C8 B0 49
.:2BF8 F0 84 D6 A5 14 69 28 85 49
.:2C00 D3 8A 20 A5 27 68 85 02 24
.:2C08 68 85 01 68 85 00 68 85 12
.:2C10 D3 68 85 D6 60 4C 27 D8 89
.:2C18 A9 00 85 0D 20 73 00 B0 38
.:2C20 03 4C F3 DC C9 C2 F0 03 80
.:2C28 4C 92 CE 20 73 00 20 F1 48
.:2C30 CE A9 CF 48 A9 E2 48 A5 86
.:2C38 15 48 A5 14 48 20 F7 D7 44
.:2C40 20 07 2D EA B0 10 C9 80 39
.:2C48 90 0C A5 14 E9 60 85 14 23
.:2C50 A5 15 E9 61 85 15 4C 16 68
.:2C58 D8 A5 99 C9 03 D0 03 4C 25
```

```
.:2C60 F6 2A 4C F5 F1 8D 0B 03 05
.:2C68 A9 6A 85 2B A9 18 8D 0A 95
.:2C70 03 A9 2B 8D 09 03 A9 A3 00
.:2C78 8D 08 03 A9 59 8D 2A 03 96
.:2C80 A9 2C 8D 2B 03 60 00 93 43
.:2C88 2A 2A 2A 2A 2A 2A 2A 2A 36
.:2C90 2A 2A 2A 2A 2A 20 43 42 75
.:2C98 4D 20 42 41 53 49 43 20 95
.:2CA0 56 32 20 2A 2A 2A 2A 2A 78
.:2CA8 2A 2A 2A 2A 2A 2A 2A 2A 36
.:2CB0 0D 2A 2A 2A 2A 20 46 41 48
.:2CB8 54 2D 34 30 20 45 58 54 02
.:2CC0 45 4E 53 49 4F 4E 20 36 46
.:2CC8 2F 31 33 2F 38 33 20 42 99
.:2CD0 59 20 50 4C 20 2A 2A 2A 35
.:2CD8 2A 0D 31 32 36 39 32 20 47
.:2CE0 42 59 54 45 53 20 46 52 75
.:2CE8 45 45 8E 0D 22 22 22 38 51
.:2CF0 C9 00 F0 03 49 FF 6A 8D 19
.:2CF8 0B 90 60 AD 8D 02 4C 5B 34
.:2D00 2E 00 10 10 0C 0C 00 A5 67
.:2D08 15 C9 E8 F0 05 A5 15 C9 86
.:2D10 84 60 A5 14 C9 48 D0 05 99
.:2D18 A9 06 4C 55 2D C9 4C D0 66
.:2D20 05 A9 04 4C 55 2D C9 4B 60
.:2D28 D0 E3 A9 03 4C 55 2D FD 66
.:2D30 A9 2C 4C 65 2C 20 D2 FF 31
.:2D38 A9 8E 20 D2 FF A9 00 8D 18
.:2D40 06 2D A9 0C 8D 04 2D 8D 63
.:2D48 05 2D A9 10 8D 03 2D A9 93
.:2D50 2E 8D 19 03 60 85 14 A9 33
.:2D58 2D 85 15 4C 0D 2D 00 A9 02
.:2D60 40 8D 27 91 38 A5 33 E5 90
.:2D68 2D 8D 3C 03 A5 34 E9 00 99
.:2D70 85 34 38 A5 34 E5 2E 8D 74
.:2D78 3D 03 A9 00 8D 3E 03 8D 80
.:2D80 3F 03 8D 40 03 F8 18 AD 19
.:2D88 3E 03 69 56 8D 3E 03 AD 35
.:2D90 3F 03 69 02 8D 3F 03 AD 53
.:2D98 40 03 69 00 8D 40 03 CE 86
.:2DA0 3D 03 D0 E2 18 AD 3E 03 60
.:2DA8 69 01 8D 3E 03 AD 3F 03 51
.:2DB0 69 00 8D 3F 03 AD 40 03 52
.:2DB8 69 00 8D 40 03 CE 3C 03 82
.:2DC0 D0 E2 D8 A2 00 A0 04 BD 65
.:2DC8 3E 03 20 E1 2D 99 DA 2C 82
.:2DD0 88 30 1C BD 3E 03 20 E6 28
.:2DD8 2D 99 DA 2C 88 E8 4C C7 03
.:2DE0 2D 29 0F 09 30 60 29 F0 35
.:2DE8 4A 4A 4A 4A 09 30 60 A5 14
.:2DF0 38 85 34 A5 37 85 33 60 41
.:2DF8 20 2C 2A A9 1E 20 52 2E 77
.:2E00 60 20 BD 28 AD 15 03 C9 55
.:2E08 2A F0 13 C9 EA F0 0F AD 64
.:2E10 14 03 8D CC 2A AD 15 03 07
.:2E18 8D CD 2A 4C 28 2E A9 EF 58
.:2E20 8D CC 2A A9 EA 8D CD 2A 78
.:2E28 AD 19 03 C9 FE F0 0D AD 82
.:2E30 18 03 8D 6D 2A AD 19 03 20
.:2E38 8D 50 2D 60 A9 47 8D 6D 52
.:2E40 2A A9 2E 8D 50 2D 60 A9 88
.:2E48 D2 8D 16 03 A9 FE 8D 17 63
.:2E50 03 00 20 3A 2D A9 3F 8D 11
.:2E58 27 91 60 C9 04 D0 07 AD 73
.:2E60 21 91 C9 FD D0 F9 4C BC 53
```

LOADER PROGRAM

```
•10 POKE36879,8:PRINT"{SC}{GN}PLEA
SE WAIT ........
•20 PRINT"{CD}VIC 40 IS LOADING FR
OM":D=PEEK(186)
•30 PRINT"FROM THE STORAGE UNIT.{C
D}{WH}"
•40 PRINT"LOAD"CHR$(34)"VIC 40"CHR
$(34)","D",1"
•50 POKE198,10:FORI=631TO640:READA
:POKEI,A:NEXT
•55 IFD=1THENPRINT"{CD}{CD}{CD}{CD
}"
•60 CLR:PRINT"{CD}{CD}{CD}{CD}POKE
36879,27:POKE11881,0:POKE43,106:P
OKE44,46:NEW"
•90 PRINT"{HM}{CD}{CD}{CD}{CD}";:E
ND
•100 DATA13,13,83,89,83,56,56,52,4
9,13
```

BUG REPELLENT LINE CODES FOR LOADER PROGRAM

```
LINE # 10:IH        LINE # 55:HF
LINE # 20:NF        LINE # 60:LM
LINE # 30:HO        LINE # 90:DD
LINE # 40:AG        LINE # 100:IG
LINE # 50:MG        LINES: 9
```

DEMO PROGRAM

```
•1 CLR
•2 SYS11768:POKE36879,8:GOSUB40:PO
KE36878,7:POKE37159,37:CLR
•3 DATA1,6,9,5,1,10,13,3,2,8,11,7,
8,0,1,4
•4 DATA8,1,5,7,8,6,15,3,8,16,17,5,
8,18,19,4,9,1,5,3,9,6,9,7,9,10,13
,5,9,14,19,7,10,8,11,5
```

IMPORTANT
Before typing in an *Ahoy!* program, refer to the first two pages of the program listings section.

```
•10 PRINT"{HM}{CD}{CD}{CD}{CR}{CR}
{CR}{CR}{CR}{CR}{CR}{CR}{CR}{CR}{
CR}{CR}{CR}AHOY! MAGAZINE"
•15 PRINT"{CR}{CR}{CR}{CR}{CR}{CR}
{CR}{CR}{CR}{CR}{CR}{CR} TTTTT TT
TTTTTT
•20 PRINT"{CR}{CR}{CR}{CR}{CR}{CR}
{CR}{CR}{CR}{CR}{CR}{CR}     PRESE
NTS
•25 PRINT"{CR}{CR}{CR}{CR}{CR}{CR}
{CR}{CR}{CR}{CR}{CR}{CR}{CR}{CR}{
CR}{CR}TTTTTTTT
•31 PRINT"{CD}{CD}             I U  I
  UCI   I U UCI
•32 PRINT"                B B  B  B
B B B B
•33 PRINT"                B B  B  B
ZC+ B B
•34 PRINT"                B B  B  B
  B B B
•35 PRINT"                JCK  J  JCK
  E JCK
•36 PRINT"{CD}{CD}{CD}{CD}{CD} A {
RV}40-COLUMN{RO} SOFTWARE EXTENSI
ON FOR THE
•37 PRINT"{CD}   COMMODORE VIC-20
PERSONAL COMPUTER.
•38 PRINT"{CD}{CD}{CR}{CR}{CR}{CR}
{CR}{CR}{CR}(C) 1983 AHOY! + PETE
R LOBL{HM}":GOTO400
•40 Q=1:CO=37888:RO=20:PRINT"{SC}{
WH}"CHR$(142)
•50 READA,B,C,D:IFA=10THENQ=0
•60 FORI=CO+RO*A+BTOCO+RO*A+C:POKE
I,D:NEXT:IFQTHEN50
•70 RETURN
•400 POKE198,0:FORJ=1TO500:GOSUB40
000:GETR$:IFR$=""THENNEXT
•420 CLR:PRINTCHR$(14)"{SC}{WH}{CD
}{CD}THE VIC 40 INCLUDES BOTH OF
COMMODORE'S
•430 PRINT"{CD}STANDARD CHARACTER
SETS AND IT GIVES
•440 PRINT"{CD}YOU THE OPPORTUNITY
 TO CREATE YOUR OWN
•450 PRINT"{CD}CHARACTERS BESIDES.
  ALSO, YOU CAN MIX
•460 PRINT"{CD}FONTS ON THE SCREEN
 AT ONCE TO CREATE
•470 PRINT"{CD}DISPLAYS NEVER-BEFO
RE SEEN OR HEARD OF.
•480 PRINT"{CD}{CD}VIC 40 HAS 256
PROGRAMMABLE CHARACTERS
•490 PRINT"{CD}AVAILABLE TO YOU IN
 WHICH VARIOUS FONTS
•500 PRINT"{CD}CAN BE CREATED. IF
YOU NEED A FULL
•510 PRINT"{CD}CHARACTER SET THEN
128 PROGRAMMABLES ARE
•520 PRINT"STILL AVAILABLE (PLUS T
HE BIT-MAP!!) .
•550 A=3000:GOSUB30000:PRINT"{SC}{
CY}";:FORI=1TO40:PRINTRIGHT$(STR$
(I),1);:NEXT
•560 FORI=2TO24:PRINTRIGHT$(STR$(I
),LEN(STR$(I))-1):NEXT:PRINT"25{H
M}{CD}{CD}{CR}{CR}{CR}";
•570 CLR:PRINT"   VIC 40 GIVES YOU
 A 40*25 SCREEN."
•580 A=0:PRINTCHR$(142):FORJ=32934
TO32934+336STEP40:FORI=0TO30:POKE
J+I,I+A:IFI+A>254THEN595
•585 NEXT:A=A+31:NEXT
•595 CLR:PRINT"{CD}{CD}{CD}{CD}{CD
}{CD}{CD}{CD}{CD}{CR}{CR}{CR}{CR}
{CR}{CR}{CR}{CR}{CR}{CR}{CR}{CR}{
CR}{CR}WITH FULL UPPER CASE !!!!!
"CHR$(14)
•605 FORJ=32918+456TO32918+792STEP
40:FORI=0TO30:POKEJ+I,I+A:IFI+A>2
54THEN630
•610 NEXT:A=A+31:NEXT
•630 CLR:PRINT"{CD}{CD}{CD}{CD}{CD
}{CD}{CD}{CD}{CD}{CR}{CR}{CR}{CR}
{CR}{CR}{CR}{CR}{CR}{CR}{CR}{CR}{
CR}{CR}AND FULL LOWER CASE !!!!!{
HM}
•650 FORI=1TO300:NEXT:C=103:GOSUB1
1100
•660 PRINT"{SC}{CD}VIC 40 CAN EMUL
ATE PET BASIC SOFTWARE
•670 PRINT"{CD}BECAUSE IT ALLOWS F
OR THE SCREEN AND
•680 PRINT"{CD}SOUND LOCATIONS TO
MOVE.  VIC SOUNDS
•690 PRINT"{CD}AND GRAPHICS WILL S
TILL WORK BUT NOW YOU
•700 PRINT"HAVE THE ADDED DIMENSIO
N OF ALL THAT PET
•710 PRINT"SOFTWARE CURRENTLY AVAI
LABLE.":A=2000:GOSUB30000
•720 PRINT"{CD}{CD}{YL}VIC 40 ALSO
 ALLOWS FOR HI-RES GRAPHICS
•730 PRINT"{CD}TO BE MIXED WITH A
NORMAL SCREEN. THIS
•740 PRINT"{CD}BIT-MAP COVERS THE
```

```
ENTIRE 160*176 HI-RES
750 PRINT"SCREEN AND PLOTTING TO
IT IS MUCH EASIER
760 PRINT"THAN THE NORMAL ACCEPTE
D WAY OF PLOTTING
770 PRINT"POINTS. WITH THE ADDED
MACHINE CODE
780 PRINT"{CD}ROUTINES YOU'LL BE
CRANKING OUT DISPLAYS
790 PRINT"YOU THOUGHT WEREN'T POS
SIBLE !!!":A=3500:GOSUB30000
810 PRINT"{CD}{CD}{WH}IN THE FOLL
OWING EXAMPLE NOTE HOW THE
820 PRINT"{CD}BIT-MAP CAN OVERLAY
 THE NORMAL SCREEN
830 PRINT"{CD}CHARACTERS PROVIDIN
G THE BASIS FOR ANY
840 PRINT"{CD}GRAPHIC APPLICATION
 FROM WORD PROCESSING
850 PRINT"TO GRAPH MAKING TO ADVE
NTURE GAMING.
860 PRINT"{CD}OH, BEFORE I FORGET
, THERE ARE 8 COLORS
870 PRINT"{CD}AVAILABLE TO BIT MA
P AND WRITE IN SO
880 PRINT"{CD}YOU BUDDING COMPUTE
R ARTISTS OUT THERE
890 PRINT"{CD}CAN TAKE FULL ADVAN
TAGE OF YOUR VIC 20 .":A=3000:GOS
UB30000
925 PRINT"{SC}{CD}FOR EXAMPLE:(WE
'LL START WITH MONOCHROME
926 PRINT"THEN ADD COLOR THEN ADD
 EXTENDED COLOR.)"
927 PRINT"{HM}{CD}{CD}{CD}{CD}{CD
}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD}
{CD} THIS WILL SHOW OVERLAYING WI
TH HI-RES.{CD}{CD}{CD}{CD}{CD}{CD
}
928 PRINT"{HM}{CD}{CD}{CD}{CD}{CD
}{CD}{CD}{CD}{CD}   {CD}{CL}{CL}Y=
COS{CD}{CL}{CL}{CL}{CL}(X)>"
929 PRINT"{HM}{CD}{CD}{CD}{CD}{CD
}{CD}{CD}{CD}{CD}                Y
=SIN{CD}{CL}{CL}{CL}{CL}{CL}<(X)
930 PRINT"{HM}{CD}{CD}{CD}{CD}{CD
}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD}
{CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{
CD}{CR}{CR}{CR}{CR}{CR}{CR}{CR}{C
R}{CR}{CR}{CR}Y=TAN{CD}{CL}{CL}{C
L}(X)>
933 PRINT"{HM}{CD}{CD}{CD}{CD}{CD
}{CD}{CD}{CD}{CD}{CD}{CD}{CD}{CD}
{CD} THIS WILL SHOW OVERLAYING WI
TH HI-RES.{CD}{CD}{CD}{CD}{CD}{CD
}
935 CLR:A=4320:B=7:C=176:D=2:Y=10
1:FORX=0TO157:GOSUB5000:NEXT
936 Z=157:I=.5:J=101:K=50:N=61:G=
101:H=141
937 FORX=.TOZSTEPI:Y=INT(J+K*SIN(
X/10)):GOSUB5000:Y=INT(J+K*COS(X/
10)):GOSUB5000:NEXT
939 FORX=.TOZSTEPI:Y=INT(J+K*TAN(
X/10)):IFY>-1ANDY<176THENGOSUB500
0
940 NEXT:CLR:A=1:B=7:C=1
950 FORI=1TO500:NEXT:FORI=37888TO
37888+219:POKEI,A:A=A+1:IFA>BTHEN
A=C
956 NEXT:FORI=1TO2500:NEXT:FORI=0
TO25:POKE36879,I:FORJ=1TO400:NEXT
:NEXT:POKE36879,8:FORI=1TO2000:NE
XT
1000 PRINT"{SC}"CHR$(14)"{CD}THIS
 HAS BEEN THE END OF THE DEMO FOR
"
1005 PRINT"{CD}THE VIC 40 OPERATI
NG SYSTEM.  EXPERIMRNT
1010 PRINT"TO FIND HOW IT CAN BES
T SUIT YOUR NEEDS."
1015 PRINT"WHATEVER YOU DO ON YOU
R VIC (OR PET),"
1020 PRINT"{CD}THE VIC 40 CAN DO
MORE INTERESTINGLY !!"
1066 FORI=1TO6500:NEXT:RUN
5000 Q=A+Y+INT(X/8)*C:POKEQ,PEEK(
Q)ORD^(B-(XANDB)):RETURN
11100 A$="{WH}{RD}{CY}{PU}{GN}{BL
}{YL}":I=1:B=0
11110 PRINTMID$(A$,I,1);:I=I+1:IF
I>7THENI=I-7
11120 B=B+1:FORJ=1TO3:NEXT:IFB<CT
HEN11110
11130 RETURN
30000 POKE198,0:FORI=1TOA/1.6:GET
R$:IFR$=""THENNEXT
30010 RETURN
40000 Q=6:S=2:V=37968:O=38028:FOR
I=VTOO:POKEI,INT(RND(1)*Q)+S:NEXT
:RETURN
36352
36352
```

BUG REPELLENT LINE CODES FOR VIC 40—DEMO

IMPORTANT

Before typing in an *Ahoy!* program, refer to the first two pages of the program listings section.

LINE # 1:JO	LINE # 450:MP	LINE # 720:MI	LINE # 935:LN
LINE # 2:HF	LINE # 460:LP	LINE # 730:PJ	LINE # 936:CF
LINE # 3:AM	LINE # 470:IH	LINE # 740:KH	LINE # 937:DN
LINE # 4:GE	LINE # 480:EF	LINE # 750:AI	LINE # 939:BG
LINE # 10:EN	LINE # 490:KP	LINE # 760:OO	LINE # 940:LE
LINE # 15:GK	LINE # 500:LN	LINE # 770:LC	LINE # 950:AE
LINE # 20:HF	LINE # 510:OE	LINE # 780:NK	LINE # 956:GE
LINE # 25:FP	LINE # 520:FL	LINE # 790:GP	LINE # 1000:FB
LINE # 31:LP	LINE # 550:BO	LINE # 810:CF	LINE # 1005:KO
LINE # 32:LG	LINE # 560:OK	LINE # 820:KN	LINE # 1010:EC
LINE # 33:GM	LINE # 570:KD	LINE # 830:OI	LINE # 1015:HH
LINE # 34:OL	LINE # 580:HC	LINE # 840:IO	LINE # 1020:MA
LINE # 35:OL	LINE # 585:MK	LINE # 850:FC	LINE # 1066:MG
LINE # 36:OA	LINE # 595:PP	LINE # 860:JD	LINE # 5000:OM
LINE # 37:AC	LINE # 605:AI	LINE # 870:HD	LINE # 11100:FF
LINE # 38:ID	LINE # 610:MK	LINE # 880:MH	LINE # 11110:EB
LINE # 40:GE	LINE # 630:HL	LINE # 890:CN	LINE # 11120:PG
LINE # 50:CG	LINE # 650:LG	LINE # 925:BA	LINE # 11130:IM
LINE # 60:BI	LINE # 660:IE	LINE # 926:MM	LINE # 30000:GE
LINE # 70:IM	LINE # 670:EA	LINE # 927:LB	LINE # 30010:IM
LINE # 400:ID	LINE # 680:CD	LINE # 928:NB	LINE # 40000:OH
LINE # 420:IL	LINE # 690:OD	LINE # 929:JM	LINE # 36352:AB
LINE # 430:FA	LINE # 700:IC	LINE # 930:MG	LINE # 58880:CB
LINE # 440:GG	LINE # 710:GM	LINE # 933:LB	LINES: 95

Continued from page 52
figure up, down, right, or left. You can also invert, obvert, or rotate your figure 180 degrees on the shift menu. The rotate menu also turns the sprite at an angle you specify. Negative angles move the design clockwise, and positive numbers counterclockwise.

If you activate the turn option with the f2 key, the program rotates the sprite 180 degrees and saves the rotations at each specified angle. For example, if you specify 18 degrees and use this function, the program will save 10 sprites, each rotated counterclockwise 18 degrees from the one before. This sequence is very helpful for animating sprites.

The color menu lets you choose different colors for your sprite as well as background and border color. A multicolor option accesses still more menus. Most of the monocolor functions are available in the multicolor mode, but not all. I found the color functions a little confusing, but I feel that more practice would clear the matter up. I much prefer colored sprites to the program's default dark gray.

The program loads sprite files, and saves up to 128 sprites in memory to tape or disk. You can print a sprite on a dot matrix printer, along with its data in both hex and decimal. You can load files only into an empty memory buffer, and cannot save individual sprites from the buffer without erasing everything else in memory. If you want to look at or edit sprites from magazines, the program accepts numerical sprite data as well as your drawing.

Beginners who want to use *Sprytebyter* should expect to study how sprites work from some other source. Even experienced programmers may find the multitude of interconnected menus confusing at first, especially since the manual is difficult to use. However, the functions available make *Sprytebyter* highly versatile, especially for animating sprites. The program takes some effort to learn, but the results are worth the effort.

Microtechnic Solutions, Inc., P.O. Box 2940, New Haven, CT 06515 (phone: 203-389-8363).

—Annette Hinshaw

THE SPREAD SHEET ASSISTANT with THE GRAPHICS ASSISTANT

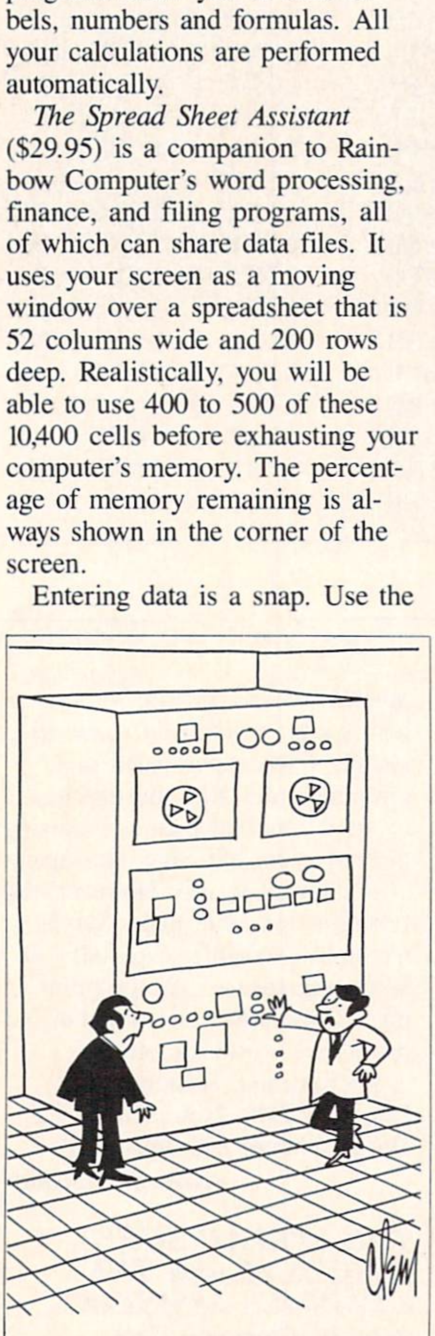
Rainbow Computer Corp.
Commodore 64
Disk

Anyone who has spent long

hours with pencil, ledger sheet, and adding machine will have his life changed—at least his working life—by switching over to a computer with a spreadsheet program. With cells representing the rectangular boxes of ledger sheets, these programs allow you to enter labels, numbers and formulas. All your calculations are performed automatically.

The Spread Sheet Assistant ($29.95) is a companion to Rainbow Computer's word processing, finance, and filing programs, all of which can share data files. It uses your screen as a moving window over a spreadsheet that is 52 columns wide and 200 rows deep. Realistically, you will be able to use 400 to 500 of these 10,400 cells before exhausting your computer's memory. The percentage of memory remaining is always shown in the corner of the screen.

Entering data is a snap. Use the cursor keys to move to the cell where you want the data and type away. In formulas, you can use the four standard arithmetic operators as well as exponentiation and one special function to total a range of cells. Each cell is referred to by its columns (A to AZ) and row (1 to 200). Be careful not to enter formulas incorrectly—whenever I left out a multiplication sign (*), the program crashed.



"I think I've found your trouble. It's upside down."

When your spreadsheet grows larger than the screen, you can jump to any cell with a single command or move any direction in full-screen increments. If you're punching in a long list of numbers, you can set the cursor to jump automatically to the next row or column when you hit return. See a mistake? You are allowed to edit without having to retype the whole entry. For long headings, there is a superb feature that lets you type across cell boundaries.

The Spread Sheet Assistant lets you add or delete rows and columns to your heart's content. All the formulas you have already entered are automatically adjusted. If you want the same formula in the February through December columns that you just entered for January, it's an easy matter to duplicate that formula right across the row instead of retyping it eleven times.

Each cell can be formatted to show data as an integer, with two decimal places ($), or right or left justified. If numbers are too long to display in the nine character wide columns, a string of asterisks will appear. Numbers with many decimal places will fill the column, running up to the last digit in the previous column. A global feature lets you change column width, set the format of all the cells, calculate by row or by column, and recalculate only on command rather than automatically. This feature would never work for me.

Eventually, you will want hard copy. Not only can you configure this program to work with almost any printer, but you can also send special print commands each time you print a page. You must specify a block of cells that will fit on the printed page; if your sheet is bigger than this, individually print several pages. There is also an option to print the formulas you have entered.

While *The Spread Sheet Assistant* is running, you can format disks and delete or overwrite old files. The disk directory is always a keystroke away, but only one filename at a time is visible. If you'd like to merge two spreadsheets, just load one on top of the other. Anyplace they use the same cell, the second sheet takes precedence over the first. This includes any cells in the second sheet that were cleared; even though blank, they will wipe out data in the first sheet.

Don't worry if this sounds like a lot to remember. The documentation—including a tutorial, reference guide, and quick-reference page (not card)—is good and is supplemented by the help screen available from the program. If you want to save data for use by other *Assistant* programs, however, you will have to experiment. The documentation for this function does not match the program.

The second program in this package is *The Graphics Assistant,* which turns your data into bar charts, line graphs, or pie charts and allows you to dress these up with free-form graphics.

To create a bar chart, you enter a title, which will be centered over your work, and labels for the vertical and horizontal axes. Give maximum and minimum vertical

values and the number of vertical ticks (labels between those values). Don't use more than twelve ticks or they will partially overwrite each other. Each of the up to 30 bars also gets its own three-letter label. If your chart has few bars, the program will make them wide so the display is still full screen. Once you have entered and edited all your data pairs, you can save them on disk. Next your chart is drawn and you have the option to save it on disk too.

What is the difference between these two saved files? The first is a data file; when it is reloaded, you can edit the data as well as the chart's labels and values. The second is a saved screen that loads much faster than your original data can be redrawn—about 30 seconds compared with three to ten minutes.

Line graphs work the same as bar charts except that you enter values, not bar labels, for the horizontal axis. Additional lines can be superimposed on your line graph, each one twice as wide as the last. The eighth line will be as wide as a character is high.

Oval-shaped pie charts can have up to ten segments with labels outside the pie. Segments appear in the order you enter them, counterclockwise, beginning at three o'clock. You are not allowed to save pie chart data or to convert data from other *Assistant* programs into pie charts.

Any chart can be dressed up with free-form graphics. You can draw solid or dotted lines, type normal or reverse video characters, erase characters, and change colors. Reverse characters allow you to put labels in pie segments and on bars. Be careful not to leave the reverse character toggle on; your next chart will have nothing but reverse characters. The erase feature works well for the screen display, but may leave odd marks on your printed chart.

Since no directory is available, loading files from disk is an experience; you can spend a long time guessing what you called a particular chart. You will have to exit the program to format a data disk. That's okay—unless you just finished a fancy chart and can not save it.

Most program options are selected from a menu; you are prompted through the rest of the process. Like the manual for *The Spread Sheet Assistant,* the documentation is well laid out. *The Graphics Assistant,* however, only works with graphics printers: Okidata 92, Commodore 1525, and Epson. Printing is good, with an occasional blank spot. Either quarter- or full-page printouts are possible. Think how nice it would be to incorporate those graphs into reports with your word processor.

Rainbow Computer Corporation, 409 Lancaster Avenue, Frazer, PA 19355 (phone: 215-296-3474).

—Richard Herring

THE VIC 40

Continued from page 65

on your VIC, sounding quite similar to the PET speaker. Consult the memory map in the next section of this article for all of the PET/CBM POKE's which the *VIC 40* will accept.

VIC 40 *MEMORY MAP*

All locations are in decimal unless preceded by a dollar sign ($), in which case they are in hex (base 16).

4096-4316	Real screen in RAM, holding locations of programmable characters (which make up the hi-res bit map).
4320-7839	Hi-res screen made up of programmable characters aligned in incrementing columns instead of the often-used row method.
7840-8839	REAL 40×25 screen which is emulated by BASIC at 32768 by the *VIC 40*'s I/O routines. Use STA or JSR $FFD2 to access this screen from ML.
8840-8959	Subroutines for *VIC 40.*
8960-9855	This is the stored programmable characters section. To create custom characters, changes must be made in this area. Characters are stored 2 per byte, with the order of characters following the same order that the standard CBM ones have.
9856-11880	Main *VIC 40* program.
11881-up	Free BASIC (or ML) RAM. You have 4500 bytes free with an 8K expander, plus another 8192 bytes for every 8K board you add.
IRQ VECTOR-10932 ($2AB9)	
PET/CBM POKE's & PEEK's	
32768-33767	Screen RAM.
59464	PET/CBM Sound Frequency. Higher values generate lower tones.
59466	Selects timbre of note.
59467	Controls volume and turns speaker on or off.
59468	Flips character set from only upper case with all graphics to entire alphabet with limited graphics characters.

SUMMARY

This is a short reminder of what the four parts of the *VIC 40* program listing do, and which ones are actually needed to RUN *VIC 40.*

VIC 40 Entry: used only to enter the hex digits into the computer. After entering the digits, a version of *VIC 40* is created for you on either disk or tape, ready to ultimately LOAD and RUN.

VIC 40 Hex: the *VIC 40* in hex format. Use *only* for entry with the *VIC 40 Entry* program. After entering *VIC 40,* this is not used again.

VIC 40 Loader: actually boots the *VIC 40* system. Must be located right before the main *VIC 40* program on tape.

VIC 40 Demo: a demonstration of some of the power of the *VIC 40.* It is by no means a comprehensive study of all the program's power, but it can point you in the general direction for grander things. Study it to gain insight on color control, simple hi-res plots, and overall character mixing. Feel free to use any of the routines within it in your original programs.

OVERALL

As I have shown, *VIC 40* is a powerful addition to your library of software. Whether you use it to run PET software, to create hi-res text/graphics adventure games, or to do serious word processing, the *VIC 40* can take the insanity out of using the normal 22 column display. Now your VIC can compete with its newer brother, the 64, as well as converse amiably with the older PET/CBM's. The better of both computers can be found in your little old VIC 20, the computer that started the whole mass-marketed microcomputer revolution. Have fun with this program, and please remember to experiment to find out how *VIC 40* can meet the best of your specific needs! □

SEE PROGRAM LISTING ON PAGE 80

BIT MAPPED GRAPHICS

Continued from page 18

mode is stored in screen memory. Each byte of screen memory contains information for two colors. Multicolor mode adds and stores the information for a third color in color memory. In addition, a common background fourth color is available in multicolor mode. □

REFERENCES

1. *Commodore 64 Programmer's Reference Guide.*
2. *Commodore 64 Color Graphics: A Beginners Guide* by Shaffer & Shaffer Applied Research & Development; The Book Company.
3. *Commodore 64 Color Graphics: An Advanced Guide* (see no. 2).

RUPERT REPORT

Continued from page 22

Line 125 prevents any J2 value greater than 10 from being used, aside from 16 which was already tested. Line 130 reads the keyboard to see if any key has been pressed. If so, line 140 assigns that new key to CH$, so the printed character is changed by pressing any key. Lines 150 and 160 print the cursor at the current position then back up and print the selected character at the same place.

A pause is added to the end of each line, so that both the cursor and the character images are discernible. You may change the values or delete the pauses entirely. Line 170 slows the whole process down so that you have better control over the positioning of the cursor. Again, the length of the pause is determined by your application. Finally line 180 moves the cursor in the direction selected by the joystick.

Lines 190 and 200 check to see if the cursor is about to move off the screen at the right or at the bottom. Location 214 tells the current cursor row (0 through 24), and location 211 tells the current cursor column (0 through 39). If the cursor is below row 23 or beyond column 38, it is moved back. Line 210 brings us back to the reading of the joystick in line 110, and the process is repeated.

You should be able to customize every aspect of this program to suit your needs. If you want to define additional values for C$, you must use a DIM statement at the beginning of the program. You should also delete line 125. The maximum value of J2 occurs when the stick is pulled back and to the right (a value of 10), and when the fire button is pushed at the same time (an additional 16). Consequently DIM C$(26) would allow you to define something for every possible joystick and fire button combination. Many values of J2 are not possible, and the corresponding values of C$ can be left undefined.

As usual, the rest is now up to you. Think of the joystick as an extension to your computer. It should be as easy to make your programs communicate with the joystick as it is with the keyboard. □

The Emerald Elephant of Cipangu

Continued from page 57

13—Dungeon	**14—Treasure Room**
N=55	N=50
S=55	S=50
E=55	E=50
W=55	W=50
U=9	U=50
D=55	D=50
I=50	I=50
O=50	O=50

A PLACE FOR EVERY THING

It isn't an adventure if all you do is wander around aimlessly, trying not to get lost. You can do that in your local shopping mall. You have to be able to pick things up and carry them with you, so you can use them when you need to.

So let's create (of course) another table, the Thing Name Table. It will be contained in the variable TN$*(n)*, and the items are listed in this order:

A GOLDEN KEY
A LOAF OF RYE BREAD
A BUTCHER KNIFE
A POCKET SUNDIAL
A DELICATELY EMBROIDERED PILLOW
A LIMP, MOTIONLESS MOUSE
A PARCHMENT CODEX
AN ORANGE RIND
A FLAGON OF ALE
THE EMERALD ELEPHANT OF CIPANGU

Remember—as with all tables, the order of this list is as important as its contents. The golden key, for instance, is the first position, is Thing number 0; the

emerald elephant is Thing 9.

Each Thing must be able to be picked up and carried by the player, and then set back down in any room. Therefore, we'll need a Thing Location Table, contained in the variable TL*(n)*. TL(0) will contain the number of the room where Thing 0 is located. At the beginning of the game, Thing 0, the golden key, is located in room 3, the Ledge East of Castle; therefore the value of TL(0) is 3.

If the player is in room 3 and types the letter T, for TAKE, the program searches the Thing Location Table to find the first Thing located in the present room (PR=3):

```
FOR I=0 TO 9:IF TL(I)=PR THEN TL(
I)=0
```

What the program does is change the location of Thing 0 to room 0. Room 0 is not one of the rooms in the castle. Instead, it is a false room which represents the player himself. Any item that is in the player's possession is considered to be located in room 0.

Whenever the player enters a room, the program must again search the Thing Location Table to see what items are in the room and display their names on the screen.

When the player chooses to Leave and item that he has, the program changes the value of that entry in the Thing Location Table from 0 to the present room number.

(In case you're wondering why I'm using Thing instead of Object or Item, there is a reason. I like to use variable names that are linked with what they do, so that the Thing Locations are in the variable TL. This makes it easier for you to follow the programming. If I used Object Location, however, the variable would be OL. It is easy, in a program listing, to mistake an O for a 0. If you accidentally typed 0 where you should have typed O, it would cause the program to crash. For that reason I try to make your life easier and don't use O very often in variable names. For your own personal programs, which no one else will ever type, you can use O if you want.)

A PIG IN A POKE

How many things can the player carry at one time? If you want to, you can let the player carry as many things as he wants. Or you can do some intricate programming, and make it so that if the player is carrying the rye bread, he can't pick up the mouse, or if the player is carrying items totalling more than fifty pounds, he can't pick up anything else. To do that, however, you'd need to set up a Thing Weight Table, and add up the weights of all the items the player is carrying. Since we're trying to keep this program simple, that's a feature we'll pass up for

READER SERVICE INDEX

now.

We will have one limitation. If the player is carrying more than five items and tries to pick up something else, he'll automatically drop whatever he has been carrying the longest.

To do this, we'll establish the Player's Poke, the string PP$. This string will never be PRINTed. We're using a string because of the very convenient string functions LEN and RIGHT$.

The string PP$ starts out empty: PP$=""

Each time the player picks up a Thing, we'll convert the Thing number to an ASCII character, using the CHR$( ) function: PP$=PP$+CHR$*(thingnumber)*. Since Things are numbered from 0 to 9, these will be unPRINTable characters—but that doesn't matter, since we're never going to PRINT them.

If we need to know how many Things the player has in his poke, we only need to get the length of the string PP$: L=LEN(PP$). L would be the number of Things the player has.

If we want to get the names of all the Things the player has and put them in the string TT$, this routine would do it:

```
IF PP$<>"" THEN L=LEN(PP$):FOR I=
1 TO L:TT$=TT$+TN$(ASC(MID$(PP$,I
,1))):NEXT:PRINT TT$
```

First, we have to make sure the player has anything at all—PP$< >"" Then each character in the string PP$ is converted to its ASCII value, which is used as the index into the Thing Name array, TN$*(n)*, and the name is added to the string TT$.

However, there is a possible problem. What if there are six Things and their names add up to more characters than the maximum one string variable can hold? One solution is to set up a second table, the Thing Shortname Table, with the items in the same order as in the Thing Name Table and the Thing Location Table: KEY, BREAD, KNIFE, SUNDIAL, PILLOW, MOUSE, CODEX, RIND, FLAGON, and ELEPHANT. We'll put it in the array TS$*(n)*, and those are the names we'll use.

LEAVING THINGS BEHIND

In a more complex program, the player would be able to specify which of the items he's carrying he wants to leave behind. Most games would let the player say DROP FLAGON. Since this game allows only single-keystroke commands, the Leave command automatically drops the Thing the player has had the longest.

It's easy to figure out which is the earliest Thing picked up, because it will be the leftmost character in the string PP$. So to Leave the oldest Thing, this routine would do nicely:

```
TL(ASC(LEFT$(PP$,1)))=PR:PP$=RIGH
T$(PP$,LEN(PP$)-1)
```

The first statement gets the ASCII value of the leftmost character in PP$ and uses it as the index into the Thing Location Table; the location of that item is set to the present room number, PR. The second statement strips off the leftmost character by saying the PP$ now should contain all the characters it contained before, counting from the right, minus the last or leftmost one.

Once these routines are working, you can add as many Things to the program as you'd like. Just DIMension all the Thing arrays—TL,TN$, and TS$—to the number of items you're going to have, add the new values in the set-up routines after line 1000, and make sure that all Thing Table searches, which now begin FOR I=0 TO 9, are changed from 9 to the number of the last item in your list. Remember to assign room numbers to every Thing in the TL array, since any item with a location of 0 is in the player's possession.

OTHER COMMANDS

We've already taken care of Take, Leave, and Got?; the remaining commands are much easier.

The Quit command gives the player one last chance to change his mind about quitting, then ENDs the program.

The Back routine acts like another directional command, except that instead of using the Room Di-

rection Table, it keeps track of what room the player was last in, and automatically goes back there. This is done by always using the variable XR to hold the number of the last room visited. Whenever the player is about to move to another room, XR is set to equal PR and PR is then changed to the new value. To move back, PR is set to the value of XR, and then XR is set to the former value of PR.

THE PEER COMMAND

The Peer command is the simplest of all—yet about half of the program consists of the routine to handle it. This is because Peer uses the present room number, PR, as the index into a jump table. For each of the fourteen rooms there is an extended description that gives more details about what can be seen at that location. And that can add up to a lot of words. Yet once the jump is made, the routine consists almost entirely of PRINT statements.

Don't let the simplicity of the command fool you. Since your words control the player's experience of the game, it is vital that your writing be as clear and interesting as possible. Each description should make the scene come to life, so the player can easily visualize where he is.

At the same time, this is the big memory eater. If you get logorrhea and fill the screen with detailed descriptions of every room, you won't be able to fit as many rooms into the computer. Since we are working with only 14 rooms, I felt free to be a little too, uh, eloquent; with a long game, you'll have to be as concise as possible. This doesn't mean that you won't have at least a few long descriptions. It just means that you won't have a long description for every room.

One thing that helps ease the need for long descriptions is clever naming. Ever since the original *Adventure*, there have been room names like Swiss Cheese Room, for a cavern chamber with lots of holes and tunnels going off in every direction, or Ice Room, or Bat Room, or Low Room. These names often tell everything essential about the room, and the description doesn't have to be too complete.

Still, don't get *too* concise, or it will start to feel like a form-letter. You never want to let the player get the feeling that one room is pretty much like every other room, and the only thing that matters is the objects he finds there. Then the story element is virtually gone.

HAVE I BEEN HERE BEFORE?

Most adventure games have an automatic Peer function. The first time the player comes to a room, the full description is PRINTed, as if the player had commanded it. After that, the full description is given only if the player asks for it.

This is a simple matter. We just set up a Visit Table with as many elements as there are rooms in the game. The table is held in the array VS*(n)*.

At the beginning of the game, all the elements of the array are set to 0. Then, every time the player enters a room, if that room's entry in the Visit Table is zero, it is set to 1 and the full description is PRINTed. Whenever that room is visited in the future, its entry in the Visit Table will *not* be zero, and so the full description will not be PRINTed.

DID I WIN?

Another automatic function is a check every time through the main loop to see if victory conditions have been met. In our game, victory occurs when the player stands outside the Castle Main Gate with the Emerald Elephant of Cipangu in his possession. The test is this simple:

```
IF PR=1 AND TL(9)=0 THEN 940
```

If the present room is 1—Castle Main Gate—and the location of Thing 9, the elephant, is 0—in the player's possession—then go execute the victory routine, which PRINTs the final message and ENDs.

WHAT THIS GAME DOESN'T HAVE

Since "The Emerald Elephant of Cipangu" is designed to be a simplified example program, there are a lot of things that it doesn't do. Even in the same 14-room castle, there are many possible complications.

We could have added commands, so that the player could choose to fight the ghost knight in the lookout tower, or ride one of the ghost horses in the stable.

We could allow the possibility of real disaster, so that if the player chose to go east while on the edge east of the castle, he would drop to his death.

Or if the player chose to go north from the main gate or west from the meadow, he might get lost in the forest; or we might add a maze to the dungeon, just to make things more complex. There are two ways to create a maze. One is to add many rooms, each of which has the same name, like "DUNGEON CORRIDOR" or "IN THE FOREST." Then the Room Direction Table can be really confusing. For instance, room 20 and 21 might be above each other—going upward from one always leads to the other. Then the only way the player can find his way is to drop something in the first room, so that he knows when he has returned to the starting point of the maze.

The area under the stable could have been a secret passage that led to the grove outside.

We could allow the sundial to tell the time, so that the player could only leave the castle at noon.

The player could get hungry and weak if he didn't eat the rye bread in the kitchen. Or the rye bread could poison him if he eats it.

Some or all of the Things could be randomly located, so the player wouldn't know from one game to the next where an item might be. But remember to do this sort of thing carefully. It can put a real crimp in a game when the key to a locked room accidentally gets put inside the locked room. Games aren't much fun if they aren't winnable.

The parchment codex could be cursed, so that while he's carrying it, the entire Room Direction Table is different, and no direction leads to the place it used to lead. That would be simple enough—just make RD a *three*-dimensional array, RD*(n1,n2,n3)*. The third index would be either a 0 or 1, depending on whether the player has the codex or not.

The ghost knight could follow the player around, or wander through the castle randomly, so that the player keeps bumping into him in different rooms. This could easily be done by using the variable KR to keep track of the knight's present room, and generating random directions for the knight to go. Every time through the main loop, the program would check to see whether KR=PR; if the two variables are equal, it means the player and knight are in the same room.

This sort of complication adds immeasurably to the fun of a game. It also adds to the complexity of the program. The nice thing is that if you program carefully enough, you can get a primitive version of a program working, SAVE it, and then begin to doctor it up, debugging each new feature as you add it.



"The computer checks out okay. I don't know what the problem could be."

In fact, this program is designed to allow exactly that, with plenty of line numbers left in the main loop and space for subroutines at 200 and set-up routines before 1990.

Many of these complications would almost require you to abandon the one-keystroke command system this program uses, and start working on real parsing. While I haven't provided programming examples, the discussion of parsing in last month's column and the string commands discussed the month before should provide all the techniques you need.

NEXT MONTH: CUSTOM CHARACTERS

Before we get buried in the line-by-line commentary on the program, let me mention that next month we'll get back to the graphics side of game programming. You may remember that three months ago I used custom characters in a program that moved ships around on the screen. Next month I'll explain how to create your own custom characters for the VIC and 64 (yes, VIC owners, I haven't forgotten you!), so that you can hold entire screen displays or screen building blocks in strings and PRINT them at will.

THE NITTY GRITTY

Now let's roll up our sleeves and examine the program line by line. You might even want to tear the program listing out of the magazine so you don't have to keep flipping from this page to page 70, where the listing begins.

The program is organized in the following pattern:

0-99	Start-up Routines
100-199	Main Loop
200-299	Empty—for expansion
300-399	Movement Handlers (Directional Commands and Back)
400-499	Thing Handlers (Take, Leave, and Got?)
500-899	Text Handlers (Help and Peer)
900-999	Incidental Routines
1000-1999	Set-up Handlers

Start-up Routines

10 DIMension the arrays.
RD contains the Room Direction Table.
CW$ contains the Command Word Table.
RN$ contains the Room Name Table.
KS contains the Keystroke Table.
TN$ contains the Thing Name Table.
TS$ contains the Thing Shortname Table.
TL contains the Thing Location Table.
VS contains the Visit Table.

20 Go execute the Set-up Handlers.

25 Go randomly choose the location of the Treasure Room.

30-35 Set the starting values of the cursor position variables.
BL$ homes the cursor, blanks line 0, and returns the cursor to position to PRINT the command word.
RL$ gets to line 2, where the room name is PRINTed, and blanks two lines.
TL$ gets to line 4, where the Thing name is PRINTed, and blanks two lines
DL$ gets to line 7 for the start of the full room description.
T$ gets to line 3 and PRINTs the lead-in "WITHIN YOUR REACH IS. . ." This is used only when there is a Thing in the room.
C$ is a carriage return, CHR$(13). It is used in many of the long text passages to get from one line to the next without a new PRINT statement.

60-81 PRINT two screens worth of introduction. After each screen, jump to the keypress routine and wait for the player to press a key.

90 Set initial variable values so we begin at room 1; then jump into the

Display Handler at line 350, in the middle of the Movement Loop.

MAIN LOOP

100 Go get a keystroke from the keystroke routine at 990.

105 PRINT the command. BL$ gets the cursor in position and blanks out the old command; CW$(CM) prints the command word the player selected.

110 Check CM. If it's less than 9, it's a direction command, and jump to 300 to execute it; then skip to 130 to check for victory before starting the main loop over again.

120 Since CM was not a direction command, subtract 8 so CM will be a number from 1 to 8, and use it with the ON-GOSUB jump table. The routines are as follows:
BACK=line 390
TAKE=400
LEAVE=450
PEER=550
GOT?=470
QUIT=190
HELP and ?=500

130 If the player is at room 1 and has the elephant (Thing 9) in his possession, he has won; go to the victory routine at 940.

140 PRINT the current command again (it might have been erased) and go back to begin the main loop again.

190-195 QUIT Routine.
Give the player a second chance. If he still wants to quit, then END.

MOVEMENT HANDLERS

300 Set XX to XR, the "last room" number, and set XR to PR, the present room number. Then set PR to the new room number, as directed by the Room Direction Table.
PR always holds the *present* room, except here, where it now holds the direction where the player wants to go. XR always holds the *last* room, except here, where it holds the present room. XX is used here to hold the last room temporarily, while the program tests to see if the new PR is a valid room or an illegal movement.

310-345 Tests

310 If the player is in the Gatehouse, has the key, and the new room is the illegal movement message 56, meaning the player is trying to get through the gate, go to the special handler at 900.

315 If the *new* room is 14, the Treasure Room, then go to the special handler that decides whether the player can get in or not.

320 If the attempted room is illegal message 56 and the player is coming from the south of the castle, then the player is trying to get through the secret postern gate; if he has the butcher knife, let him through to room 5, the Kitchen.

325 If the player is trying to get back through the postern gate with the emerald elephant, go to the special routine at line 925.

330-340 Additional tests, if you add more special features.

345 Set the Description String, DS$, to ""; if the new room is an illegal-movement message, then load the message into DS$, and set PR and XR back to their previous values.

350-380 Display Handler. These lines are used as a return point by most routines in the program, since this is where the display is set up and PRINTed. The exception is the PEER handler.

350 Set TN$ to null, scan the Thing Location Table to see if anything is in the room, set TN$ to contain *all* the things that are present, separated by commas.

355 Strip the last comma off TN$.

360 Set TT$ to the "WITHIN YOUR REACH" message, unless there isn't anything here.

365 Set RN$ to the correct entry in the Room Name Table.

370 If this is the first visit to the room, set VS(PR) to 1 and go execute the Peer Routine at 550.

375 Clear the screen, PRINT the command word, get to the room line, PRINT the room name, PRINT the "WITHIN YOUR REACH" message (null if there's nothing there), get to the thing line, PRINT the thing name, get to the description line, and print whatever message is in the string DS$.

380 Go back to the main loop.

390 BACK Handler. Switch the values in PR and XR; then return to the main loop through the Display Handler at 350.

THING HANDLERS

400-420 TAKE Handler.

400 Initialize I as O.

405 For each value of I, check to see if Thing number I is located at the present room (PR). If so, add the character with ASCII value I to the player's poke, the string PP$; then set DS$ to the "GOT IT!" message and return to the main loop through the Display Handler at 350.

410 If more than five things are in the player's possession, go drop the oldest one by using the LEAVE handler at 450.

415 Add 1 to I; if I is greater than 9, it means that there was nothing in the room to take—so set DS$ to the "NOTHING HERE" message and return through 350.

420 Go back with the new value of I and conduct these tests again.

450-460 LEAVE Handler.

450 If the player has nothing, then set DS$ to the "YOU'VE GOT NOTHING" message.

455 Set LT to the ASCII value of the leftmost character in the player's poke, the string PP$. This is the Thing the player has had longest. Then set PP$ so it consists of everything that was in it before, *except* the leftmost character, or oldest Thing.

460 Change the location of LT, the Thing being left, to the number of the present room and return to the main loop through 350.

470-480 GOT? Handler.

470 If PP$ is empty, set DS$ to the "YOU'VE GOT NOTHING" message and go back.

475 Set DS$ to the lead-in message, "YOU'VE GOT:" plus two carriage returns.

480 Go through everything in the player's poke, the string PP$, and add the short name of the Thing to the DS$ string, with carriage returns in between. Then return to the main loop through 350.

TEXT HANDLERS

500-507 HELP Handler. Clear the screen, print the list of commands, wait for a keypress, clear the screen again, and return through 350.

550-565 PEER Handler. Clear the screen and, using PR as the index, jump to one of 14 possible descriptions. Then PRINT everything else that is normally PRINTed at 350, *except* DS$, and return to the main loop.

600-875 Room Descriptions.

INCIDENTAL ROUTINES

900-928 Special Event Handlers.

940-950 Victory Handler. PRINT the victory message and END.

970-986 Treasure Room Placement. Randomly select one of the legal locations for the Treasure Room.

990-995 Keystroke Handler. Wait for a keypress, then set CM to the correct value from the Keystroke Table, KS*(n)*.

SET-UP ROUTINES

1000 Set up the Command Word Table.

1050 Set up the Keystroke Table.

1100 Set up the Room Direction Table and Room Name Table.

1200 Set up illegal movement messages in the Room Name array.

1300 Set up the Thing Name Table and Thing Shortname Table.

1350 Set up the Thing Location Table.

1990 Return from the set-up subroutine.

SEE PROGRAM LISTING ON PAGE 70

BAM

Continued from page 53

free bytes on that track. Thus, for track one, the count will be a maximum of 21, For track 35, the count will be a maximum of 17. If you are familiar with hexadecimal numbers and run the *Display T & S* program on the Test Demo disk, you will notice a $15 (decimal 21) leading the group of bytes for any of the empty tracks up to number 17. The following table shows the maximum number of blocks available on each track of a 1541 formatted floppy disk.

TRACK BLOCK CAPACITY —1541 DISK DRIVE

TRACK NO.	BLOCK RANGE	TOTAL
1 to 17	0 to 20	21
18 to 24	0 to 18	19
25 to 30	0 to 17	18
31 to 35	0 to 16	17

The next three bytes of each group are the bit map for the corresponding track. For the first byte, bits 0 to 7 correspond to blocks 0 to 7 respectively. For the second byte, the correspondence is to blocks 8 to 15. The third byte has more bits than needed to complete the track. For track one, only five more blocks are available and for track 35 only one more block. Thus, only bits 0 to 4 will be used for track one. Bits 5 to 7 will always be set to zero. Similarly,

only bit 0 will be used for track 35.

The following chart illustrates the correspondence utilizing track 14 of the Test Demo disk as an example.

BYTE NO.	1	2	3	4
HEX VALUE	$11	$D7	$5F	$1F
BLOCK NO.		76543210	111111 54321098	21111 ***09876
BIT MAP		11010111	01011111	00011111

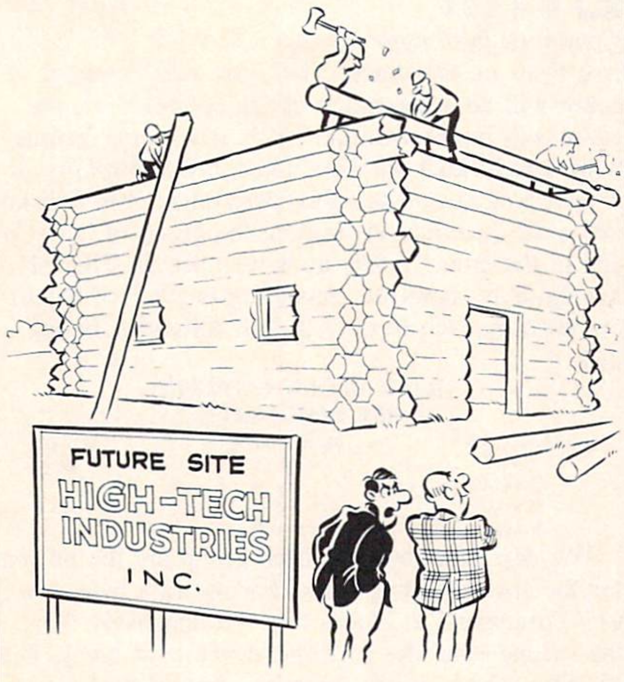
*Not used—always zero

The accompanying program will display the entire block availability map when used with the Commodore 64. Each track is laid out vertically. Thus, each horizontal row corresponds to a particular block. Track and block numbers are displayed on the top and left respectively. To use, simply load and run the program. Insert the disk under test and respond to the screen or printer prompt. If you have a Commodore printer, or an interface on the serial bus, the display may be printed out before going to the screen. The display will remain on the screen until any key is pressed. At this point, the program prompts for another disk.

The process of analyzing the BAM takes a little while using this algorithm. To let you know that the computer is alive and well, the track being analyzed is displayed on the screen.

Quite a bit can be learned by carefully experimenting with this program. Start with a blank disk. Save a file. Notice how the DOS allocates space from the center outwards. Remember, a "1" corresponds to an available block. A "0" signifies an assigned block. Save some more files. Examine which blocks are allocated when files are added to the disk. Scratch a file. Save another file. Notice the way the DOS keeps track of things. Try a save and replace. Make a careful comparison of the before and after situation for this one.

We must apologize to VIC 20 owners. There just is not enough room on the VIC 20 display to view the entire BAM without resorting to custom characters or high res graphics. Nevertheless, the program can still be used with the VIC 20 if you have a printer. Just ignore the strange looking video display. As an exercise, VIC 20 users may want to convert the video display into two parts.



"Well, if you ask me, we're starting off on the wrong foot."

HOW IT WORKS

Rather than annotate the entire listing with REMarks, we have provided a brief line-by-line explanation in the accompanying table. Most of the logic is rather straightforward. Line 280, however, may cause some head scratching. This line calculates the maximum number of blocks for a particular track. We could have coded it on several lines, in perhaps a more obvious manner. For example:

```
280 IF BL<17 THEN T=34: GOTO 290
282 IF BL=17 THEN T=29: GOTO 290
284 IF BL=18 THEN T=23: GOTO 290
286 IF BL>18 THEN T=16
290 PRINT RIGHT$...
```

We just could not resist squeezing it all in on one line. Just remember, whenever any of the expressions between parentheses is true, the result is "−1"; when false, the result is "0". Thus, "18*(BL>18) evaluates to "−18" when BL is greater than 18.

LINE NO.	
60-80	Initializes the disk, allocates a buffer and reads track 18 block 0.
90	Skips first four bytes.
100-200	Main loop, calculates free and allocated blocks.
110	Stores number of free blocks for each track in column 21 of the array SE%.
120-130	Gets three bytes representing the bit map of each track.
140-190	Decodes the bit map and stores it in array SE%, columns 0-20.
220	Selects printer.
230-390	Handles screen display.
280	See explanation in text.
400-530	Handles printer formatting.
460	See explanation for line 280.

SEE PROGRAM LISTING ON PAGE 79

COMMODARES

Continued from page 23

Problem #9-2: Never Ending? does indeed end. Once N equals 50, the statement J=N=50 assigns the value of −1 to J, since N=50 is a true statement, and true is represented as −1. Once J equals −1, the FOR-NEXT loop is completed, and the program ends. Thanks for a thought-provoking challenge from Haley Carter.

One solution for *Problem #9-3: Common Letters* is listed below. It goes through the first word letter by letter and tallies the number of matches it finds in the second word. Line 70 sets M equal to L2 if a match is found so that the FOR-NEXT loop will be terminated in line 80.

```
1 REM  PROBLEM #9-3:
2 REM      COMMON LETTERS
10 INPUT "WORD 1, WORD2";W1$,W2$
20 L1=LEN(W1$):L2=LEN(W2$)
30 FOR N=1 TO L1
40 L$=MID$(W1$,N,1)
50 FOR M=1 TO L2
60 IF L$<>MID$(W2$,M,1) THEN 80
70 CT=CT+1:M=L2
80 NEXT M
90 NEXT N
100 PRINT CT;"LETTERS OF ";W1$;
110 PRINT " ARE IN ";W2$
```

A somewhat unorthodox solution to *Problem #9-4: Letter Sorter* is shown below.

```
1 REM PROBLEM #9-4:
2 REM  LETTER SORTER
5 DIM CT(26)
10 INPUT "WHAT SENTENCE";S$
20 L=LEN(S$)
25 REM -- FILL ARRAY CT() --
30 FOR M=1 TO L
40 V=ASC(MID$(S$,M,1))-64
45 IF V>26 OR V<0 THEN 60
50 CT(V)=CT(V)+1
60 NEXT M
65 REM -- PRINT ARRAY CT() --
70 FOR N=1 TO 26
80 IF CT(N)=0 THEN 110
90 FOR J=1 TO CT(N)
100 PRINT CHR$(N+64);:NEXT J
110 NEXT N
```

A numeric array CT( ) stores a count of the number of occurrences of each letter. With every "A" that is found CT(1) is incremented; with every "Z", CT(26) is incremented; and so forth. Every letter in the sentence is counted in lines 30 through 60. Next the program prints each letter of the alphabet the number of times indicated by CT( ). If CT(4) is 5, five D's will be printed. If CT(26) is 0, no Z's will be printed. Notice in lines 40 and 100 that the nth letter of the alphabet has an ASCII value which is (n+64). ASC("A") is 65, and CHR$(65) is "A".

A solution without comment to *Problem #4-4: Random Repetition* is submitted by James Pring (Rantoul, IL). You PEEK and POKE and Boolean algebra fans should enjoy figuring out what makes this program work.

```
1 REM PROBLEM #4-4
2 REM  RANDOM REPETITION
3 REM     BY JAMES R. PRING
10 FORX=1024TO2023:POKEX+54272,PE
EK(646):Y=INT(RND(0)*10)+48:POKEX
,Y
20 IF(PEEK(X-1)ANDNOT128)=YTHENPO
KEX-1,YOR128:POKEX,YOR128
30 NEXT
40 GETA$:IFA$=""THEN40
```

A visually interesting variation on the theme of *Problem #5-4: Letter Triangle* was submitted by Paul T. Dawson (Springhouse, PA). See if you can understand his use of the ABS function.

```
1 REM PROBLEM #5-4
2 REM LETTER TRIANGLE
3 REM  BY PAUL DAWSON
10 FOR A=-25 TO 24:PRINT:PRINT TA
B(20-ABS(A)/2);
20 FOR B=0 TO ABS(A):PRINT CHR$(A
BS(A)+65);:NEXT B,A:GOTO 10
```

The *Commodares* from the June issue brought dozens of solutions. Several readers sent ultra-fast solutions to *Problem #6-1: Speed Demon* which unfortunately avoided the point of the problem. The idea was to assign 10 values to 10 variables as quickly as possible. In order to get a meaningful measure of the time required, the entire assignment routine was to be repeated 100 times. Surprisingly, more work was spent in avoiding the 100 repetitions than in speeding up the assignment statements.

Several other readers sent in proper solutions but failed to indicate the results of running their programs. Time doesn't allow me to type and run all of the programs everybody sends. Always send a printing of your output, or at least tell me in words any special properties and results of your program.

The best analysis of *Problem #6-1* came from Roland Frechette (Somersworth, NH). He measured 124 jiffies for 100 iterations of the most obvious solution

```
10 A=1:B=2:C=3:D=4:E=5:F=6:G=7:H=
8:I=9:J=10
```

He tried using FOR-NEXT loops and GOTOs without much success. Realizing that computers love binary, he tried the following:

```
10 A=1
11 B=1+1
12 C=1+1+1
13 D=1+1+1+1
(ETC.)
```

This tongue-in-cheek binary solution took 521 jiffies. The fastest solution he could find was the somewhat unexpected result:

```
10 A=1
11 B=2
12 C=3
13 D=4
(ETC.)
```

which took 121 jiffies. On my C-64, the single line 10 above was 2 jiffies slower than this one statement per line program. My times were 153 and 151 jiffies respectively.

Both of our times were put to shame by the machine language solution submitted by Mark Robin (Bloomington, MN). His routine took only 2 jiffies to perform 100 iterations of assigning values to 10 variables. That shows one of the big advantages of machine language routines.

The following program from James C. Dunavant (Gainesville, FL) is an interesting interpretation of *Problem #6-3: String Challenge.*

```
1 REM PROBLEM #6-3
2 REM  STRING CHALLENGE
3 REM   BY JAMES DUNAVANT
10 INPUT"STRING TO BE SEARCHED";B
$
20 PRINT:INPUT"STRING TO BE FOUND
";A$
30 B=LEN(B$):A=LEN(A$):IFB<AGOTO1
0
40 GOSUB500
50 PRINT:IFN=0THENPRINTB$": N = 0
":END
60 PRINTLEFT$(B$,N-1)CHR$(18)MID$
(B$,N,A)CHR$(146)RIGHT$(B$,B-N-A+
1)": N ="N:END
500 FORI=1TOB-A+1:IFMID$(B$,I,A)=
A$THENN=I:I=B-A+1:RETURN
510 NEXT:N=0:RETURN
```

His subroutine at line 500 performs the INSTRING function, and the output at line 60 highlights the substring within the string. Thanks to James and the other programmers mentioned above for their work on these problems.

The following people also submitted solutions to *Commodares* through the June issue, many of which are also very well done. Space simply doesn't permit listing all of the fine ideas these people sent. This list is in somewhat chronological order.

Dave Stevens (Auburn, ME)
Paul Lalli (McAlester, OK)
Trevor Sellar (Langham, Saskatchewan)
Bill Mallison (Rocky Mount, NC)
Geoff Williams (Sedalia, MO)
Roger Baim (Chicago, IL)
Greg Smith (Tyndall AFB, FL)
Eric Wolff (Cinti, OH)
Richard Auchenpaugh (Powder Springs, GA)
Walter Deuchler, Jr. (Aurora, IL)
Todd McMullen (Gaylord, MI)
John Kubiac (Sterling Hgts., MI)
Ron Lalonde (Inuvik, NWT)
Peter Zografos (Calais, ME)
Gary Forney (Delwein, IA)
Trevor George (Brooklyn, NY)
Ralph Juliano Jr. (Inverness, FL)
Mark Ziemba (San Angelo, TX)
D. Daniel Sabin (Salem, MA)
Jack Foley (Knoxville, TN)
Paul Lalli (McAlester, OK)
David Greenlow (Satellite Beach, FL)
John F. Adams (St. Charles, IL)
Rob Svirskas (Port Charlotte, FL)
Larry Smith (Louisville, KE)
Robert Foley (Umatilla, FL)
Philip Whitley (Greenville, SC).